

ER Modeling

with the Higher-Order Entity-Relationship Model
(In a nutshell)

Bernhard Thalheim

Brandenburg University of Technology at Cottbus, Computer Science Institute,
Universitätsplatz 3-4, 03044 Cottbus, Germany, thalheim@informatik.tu-cottbus.de

Overview

- HERM in a nutshell
- Codesign of structuring, functionality (or behavior) and interacting
- References

Extended Entity-Relationship Models

Syntax of the model

Structuring as **consistent** extension of the classical entity-relationship model

Behavior specification on the basis of generic operations forming the algebra and
on the basis of ASM semantics

Interacting of users with the information engine on the basis of their specific scope
on the database

Environment: technical context, organizational context, distribution

Views, containers for delivery of information to the user and for accepting
information from the user

Semantics via hierarchical predicate logics

can be extended by pragmatics (variety of semantics)

Pragmatism based on the codesign methodology

Constructs of the Model

Structuring as pair

Structuring := (Structure, static Constraints)

Structure as (marked) expression on constructors

Behavior as pair (Functionality, constraints)

Behavior := ((StateChange \cup Retrieval)Operations ,
dynamicConstraints)

Operations on the basis of the HERM algebra (for modification and retrieval)
views

Interacting as 4-tuple

Interacting := (StorySpace, Actors,
MediaObjects, Presentation)

StorySpace as graph of scenes and activities

Actors are abstractions of user groups

MediaObjects are used by actors and are based on views

Structuring in HERM

Structuring - extension of the ER model

strict set semantics (no pointer semantics)

1. Complex attributes, entity, relationship, cluster types

types of higher order, hierarchical schemata for representation of the structuring

2. Static integrity constraints

Basic data types - parameterized types of the DBMS

$integer := (IntegerSet, \{0, s, +, -, *, \div, \}, \{ =, \le \})$

Attribute type induced on basic data type system

Name : (*FirstNames* < (*FirstName*, *use*) >, *FamName*,
 [*NameOfBirth*,] *Title*: {*AcadTitle*} \cup *FamTitle*)

Contact : (*Phone*({*AtWork*}, *private*), *email*, ...)

DateOfBirth :: *date*

AcadTitel :: *titleType*

Entity type - product of attribute types with at least one direct identification etc.

Person : (*Name*, *Address*, *Contact*, *DateOfBirth*,
PersNo: *StudNo* \cup *EmplNo*, ...)

Relationship type via hierarchical construction

unary type: Role, specialisation

InGroup : (*Person*, *Group*,
 Time(From [,To]), Position)

DirectPrerequisite : (*hasPrerequisite: Course*, *isPrerequisite : Course*)

Professor : (*Person*, *Specialization*)

Cluster type - disjoint (labelled) union

especially for generalization

JuristPerson : *Person* $\dot{\cup}$ *Company* $\dot{\cup}$ *Association*

Group : *Senat* $\dot{\cup}$ *WorkingGroup* $\dot{\cup}$ *Association*

Constructors $\times, \cup, \{\}$ construction complete

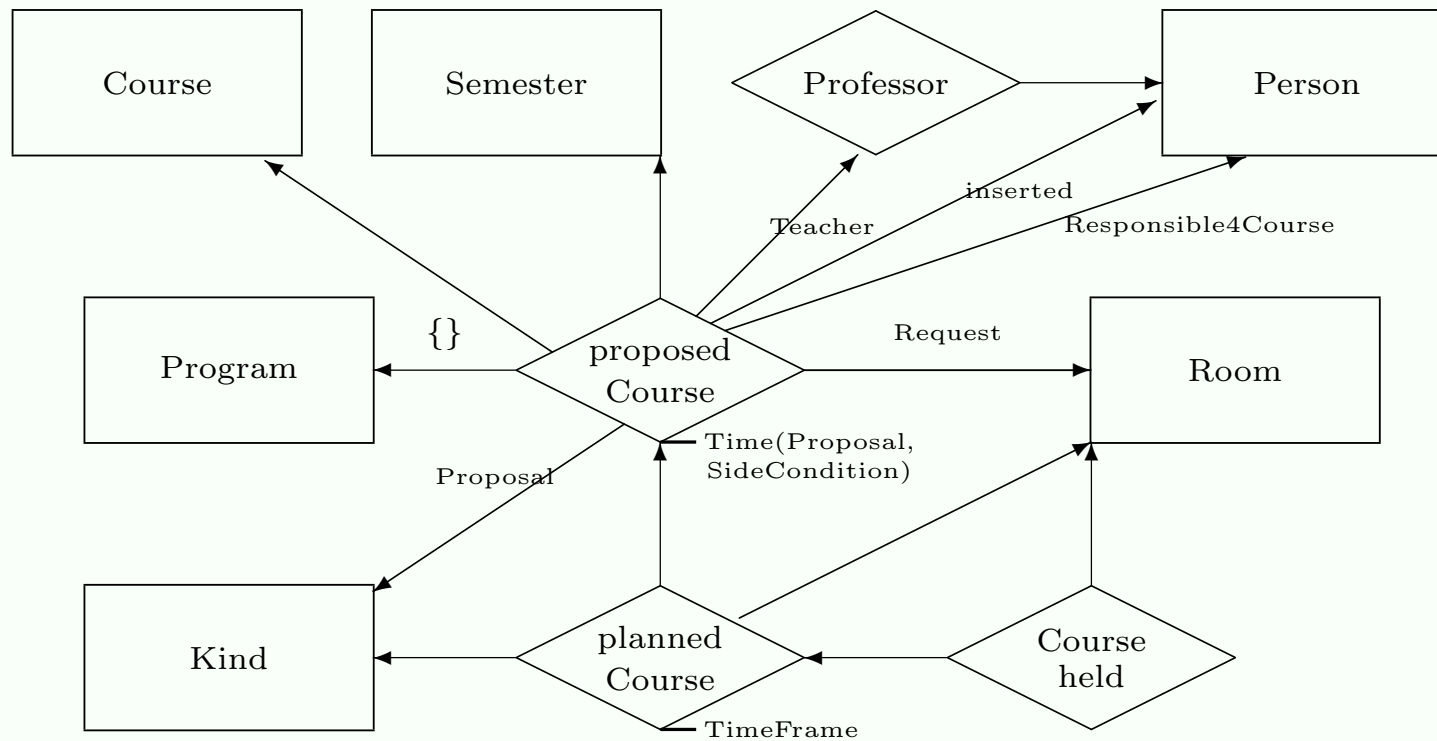
usually: $\times, \dot{\cup}, < . >, \{\cdot\}, \mathcal{P}$

with underlying type system

and generic operations

plus construction of operations through structural recursion

A Sample Schema



Static integrity constraints are formulas of the hierarchical predicate logic (with abbreviations) in “normal definition frame” (may be deontic or strikt)

Keys for identification of objects (esp. entity types)

$$\text{Key}(\text{Person}) = \{ \{ \text{PersNo} \}, \{ \text{Name}, \text{DateOfBirht} \} \}$$

Relationship types with attributs

$$\text{Key}(\text{InGroup}) = \{ \{ \text{Person}, \text{Group}, \text{Time} \} \} \quad (!?)$$

$$\text{Key}'(\text{InGroup}) = \{ \{ \text{Person}, \text{Group}, \text{Time}, \text{Position} \} \}$$

$$\text{Key}(\text{Lecture}) = \{ \{ \text{Course}, \text{Semester} \}, \{ \text{Semester}, \text{Room}, \text{Time} \}, \{ \text{Semester}, \text{Teacher}, \text{Time} \} \}$$

Keys defined by the component construction

$$\text{Name}(\text{FirstNames} \langle (\text{FistName}, \text{use}) \rangle, \text{FamName})$$

at least one key is ‘inherited’ from the component

Functional dependencies for functional associations among groups of attributes

$$\text{plannedCourse} : \{ \text{Sem}, \text{Time}, \text{Room} \} \rightarrow \{ \{ \text{Program} \}, \text{Prof}, \text{Course} \}$$

$$\text{plannedCourse} : \{ \text{Prof}, \text{Sem}, \text{Time} \} \rightarrow \{ \text{Course}, \text{Room} \}$$

$$\text{proposedCourse} : \{ \text{Semester}, \text{Course} \} \rightarrow \{ \text{Prof} \} \quad (??)$$

and other relational constraints

e.g. Domain constraints

$$\text{Semester.Description} \in \{ \text{WS}, \text{SS} \} \times \{ x/x+1 \mid x \in 1980..99, 2000..03 \}$$

Cardinality constraints are restrictions of combinatorics with (min,max)-notation

$$\text{card}(\text{DirectPrerequisite}, \text{hasPrerequisite}) = (0,2)$$

$$\text{card}(\text{DirektVoraussetz}, \text{isPrerequisite}) = (3,4)$$

not satisfiable

$$\text{card}(\text{plannedCourse}, \text{Course Sem Prof}) = (0,3)$$

$$\text{card}(\text{proposedCourse}, \text{Sem Prof}) = (3,5) (!??)$$

or

$$(0,5) (!!)$$

Classes are **sets** of (markable) objects of corresponding type

Entity class is the basic class of objects

β : ($\langle (Karl, add), (Bernhard, call) \rangle$, Thalheim, {Prof., Dr.rer.nat.habil, Dipl.-Math.}), BTU Cottbus,
 (({ +49 355 692700, +49 355 692397}, +49 355 824054),
 thalheim@informatik.tu-cottbus.de), 10.3.52, 637861)

(DBIV, Database theory, CS Programm , mandatory, 2+0+0, certificate)

Relationship classes for association of objects of component classes expanded by properties
 (through attributes)

Prof β : (637861, Database and information systems)

Senator3 β : (637861, Senat, (1995,1998), Dekan)

Senator5 β : (637861, Senat, (2000), Chair)

PrerequDBIVMain: (DBIV, DBI) ,

CourseDBIVSS02: (DBIV, SS2002, 637861, SR1, Mo. first pair)

Cluster classes for disjoint (!), i.a. virtual union of objects of the component classes

{ Senator2 π : (889721, Senat, (1998,2000), Chair),

Senator5 β : (637861, Senat, (2000), Chair),

DBIS: (Database and information systems, 637861, IfI),

CBnet β : (CottbusNet e.V., 637861, Member, Services Group) }

Classes are extended by generic operations *insert*, *delete*, *update*, *retrieve*
 may have additional methods

static integrity constraints must remain valid

inherent integrity constraint: existence of components

Representation and Storage possibly with weak value-based OID or label

either as full objects (with all properties (XML))) or as

class-separated object (similar to snowflakes) [object-relational representation]

Higher-Order Entity-Relationship-Modell

Functionality on the basis of HERM/LC

with HERM-algebra, HERM/QBE, query forms and transactions

with some kinds of dynamic integrity constraints, behavior

GCS integrity enforcement instead of rule triggering pitfalls

Interactivity in integrated form

description of dialogue scenes

(cooperation, messages, scenario, dialogue objects)

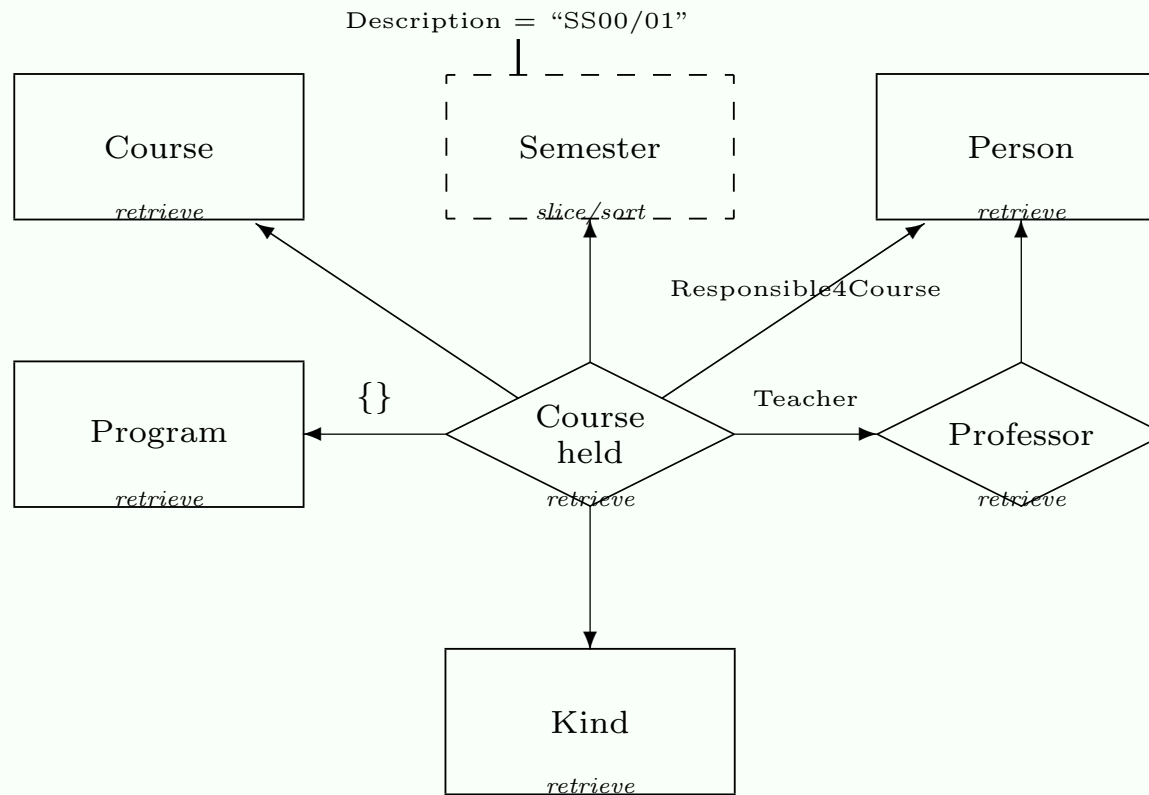
state-based

Translation and transformation methods for compilation of design into other models (logical, physical) or XML

see B. Thalheim, Entity-Relationship Modeling. Springer, Berlin, 2000

Development and engineering methods for pragmatism (see my homepage)

Views: Archiv view



XML as the display medium

XML specifications can be automatically generated out of this view

Algebraic expressions for views

Views for internet presentation as Read-Only-View

Archiv view as materialized Read-Only-View

Slice SS00/01 with Archiv.Semester := e(Semester)

for e = $\sigma_{\text{Bezeichn}=\text{"SS00/01"}}$

Archiv.Course := e(CourseHeld [Course])

*Archiv.Person := e(CourseHeld[plannedCourse[
proposedCourse[Responsible4Course : Person]]])*

Program, Kind, Professor analogous

*Archiv.CourseHeld := e(CourseHeld[plannedCourse[
proposedCourse [Course, Program, Teacher:Professor,
Responsible4Course : Person], Kind])*

Be careful: changing set of integrity constraints!

Insert view for new course proposals: als Read-Write-View with identifiable sub-views
and side conditions

with optional and mandatory components

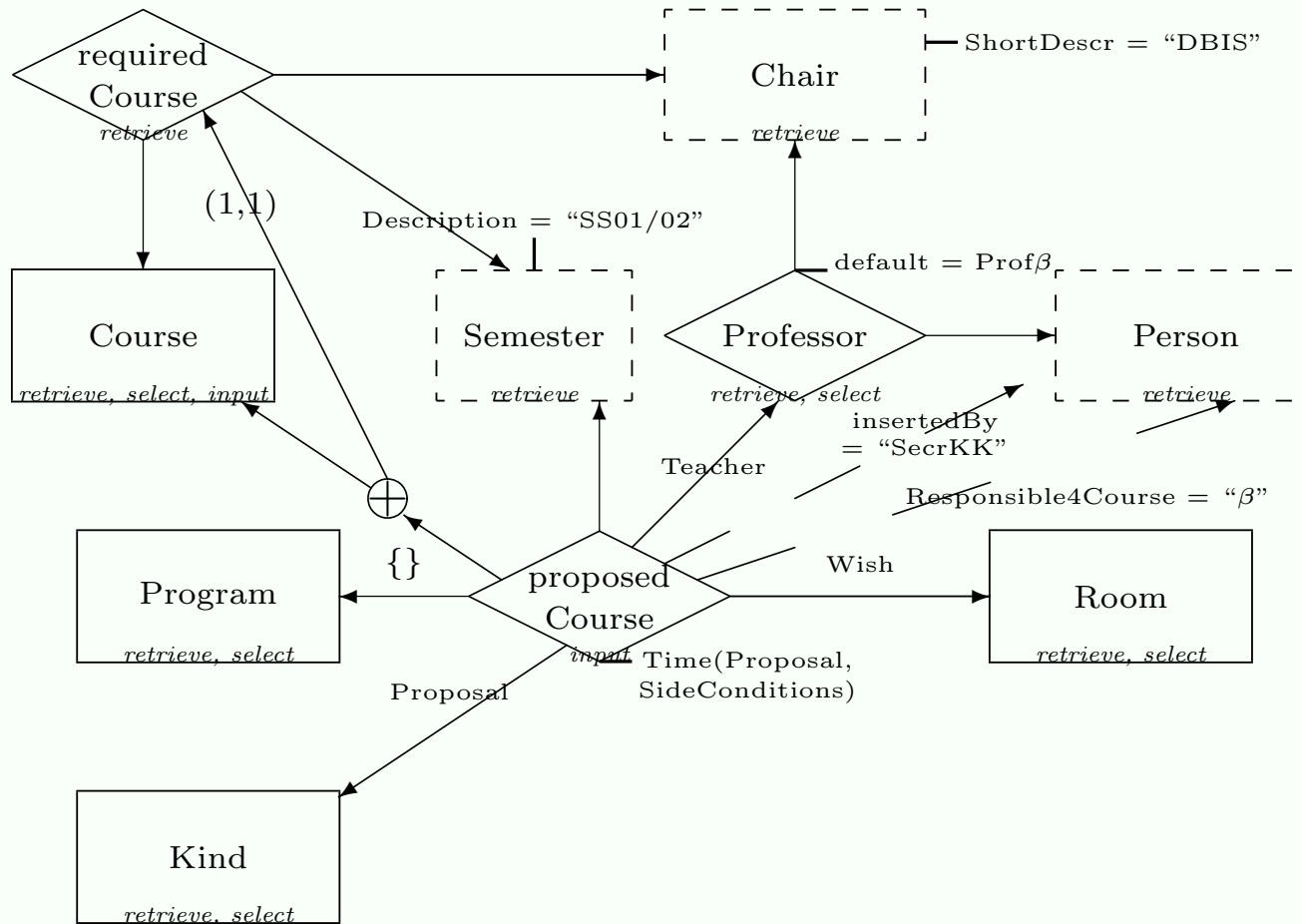
Media object schema as containers based on views with container functionality and
attached functions

representation bound by order, adhesion, cohesion of types

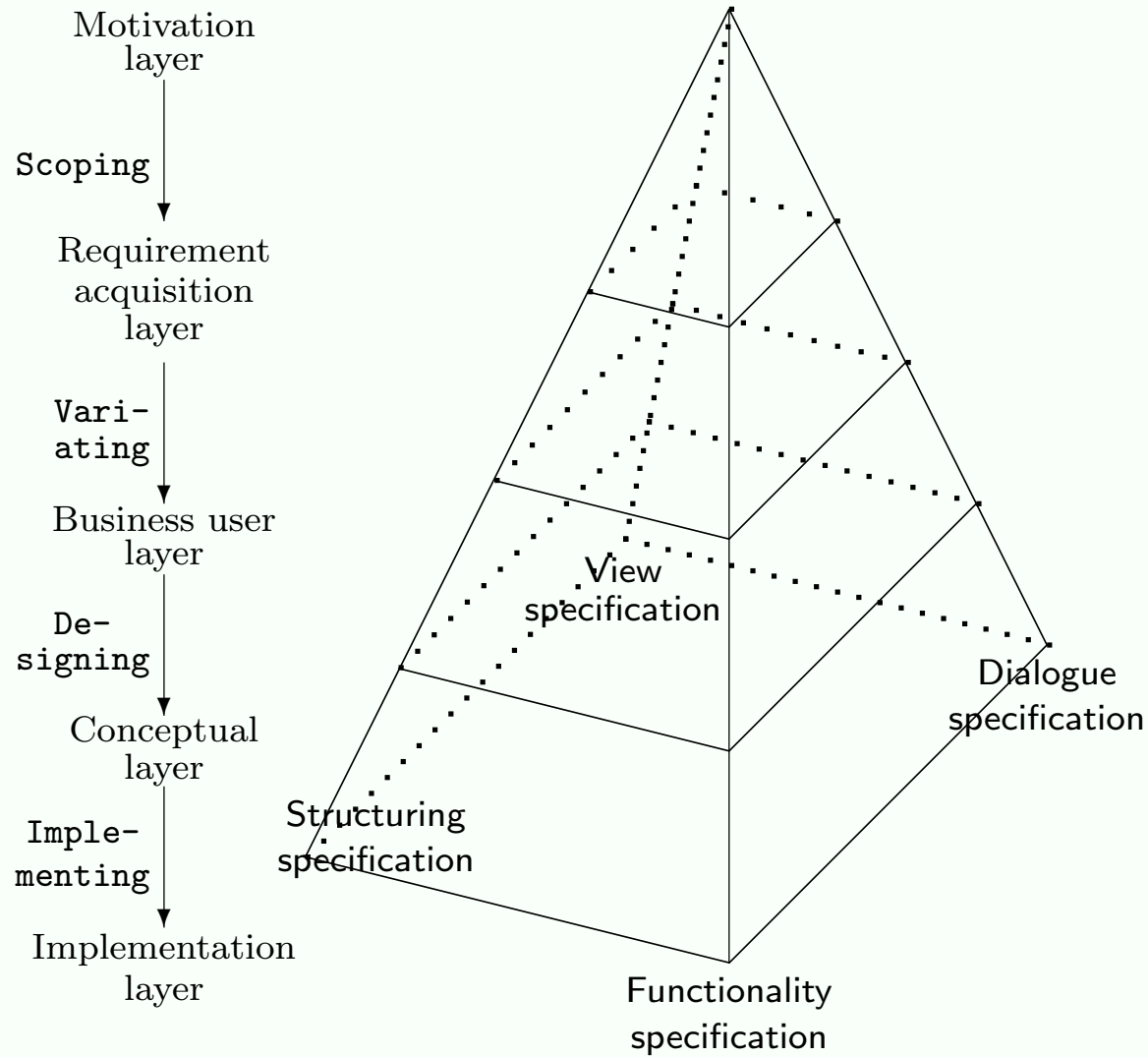
taylorred by user profile, user environment, history, channel capacity

are associated with dialogue scenes

View: Insertion view for new course proposals



Methodology: Abstraction Design Layers



Main Publications on (H)ER Modeling

- Düsterhöft, A., Thalheim, B.: SiteLang: Conceptual Modeling of Internet Sites. Proc. ER'2001, LNCS 2224, 179 - 192. *Application to webservices*
- Feyer, Th.; Thalheim, B.: E/R Based Scenario Modeling for Rapid Prototyping of Web Information Services. Proc. WWWCM'99, 253 - 263. *Application to webservices generation*
- Goldin, D., Srinivasa, S., Thalheim, B.: IS=DBS + Interaction: Towards principles of information system design. Proc. ER 2000, LNCS 1920, 140 - 153. *The theoretical foundation*
- Klettke, M.: Reuse of database design decisions. Proc. REIS'2000, LNCS 1727, 213-224. *Reuse structures and intelligently acquire integrity constraints*
- Lewerenz, J., Schewe, K.-D., Thalheim, B.: Modelling data warehouses and OLAP applications by means of dialogue objects. Proc. ER'1999, LNCS 1728, 354-368. *OLAP in a consistent, powerful and simple way*
- Schewe, K.-D.; Thalheim, B.: Towards a theory of consistency enforcement. Acta Informatica, 36, 1999, 97-141. *Instead of falling into the traps of rule triggering systems*
- Steeg, M; Thalheim, B.: Conceptual Database Application Tuning. Proc. SCI'2000, 226-231. *Tune instead of normalize*
- Thalheim, B.: Entity-Relationship Modeling - Foundations of Database Technology. Springer, Berlin, 2000. *The HERM "bible"*
- Thalheim, B.: Logics and Database Modeling. Proc. ICLP '99, MIT Press, 6-21. *The relationship to logics*
- Thalheim, B.: Codesign of database systems and interaction - Thin and consistent UML. Proc. OTS'2000, 1-17. *Codesign - the ultimate basis for best practices UML*