Data and Process Modelling

3. Object-Role Modeling - CSDP Step 1

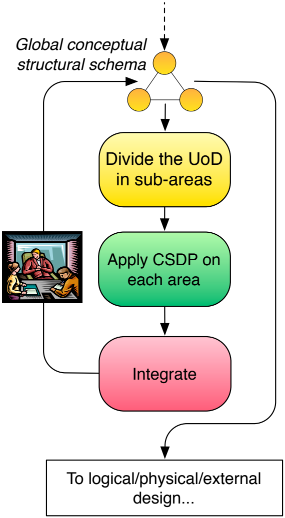
Marco Montali

KRDB Research Centre for Knowledge and Data
Faculty of Computer Science
Free University of Bozen-Bolzano

A.Y. 2014/2015

CSDP Methodology

ORM provides a Conceptual Schema Design Procedure.



1. Transform familiar examples into elementary facts.
2. Draw the fact types, and apply a population check.
3. Check for entity types to be combined, and note any arithmetic derivations.
4. Add uniqueness constraints, and check the arity of fact types.
5. Add mandatory role constraints, and check for logical derivations.
6. Add value, set-comparison, and subtyping constraints.
7. Add further constraints, do final checks.

From Examples to Elementary Facts

CSDP Step 1

Transform familiar examples into elementary facts.

- Most critical step: understanding the UoD.
- Goal: isolate relevant information to be represented in the IS.
  - Every relevant piece of information: must be elementary or derivable.
  - $\rightarrow$ Isolate each elementary fact.
    - Cannot be split into smaller units of information.
    - Simple assertion, atomic proposition about the UoD.
    - Epistemic commitment: people act as they *believed* the fact to be true.

From Examples to Elementary Facts

CSDP Step 1

Transform familiar examples into elementary facts.

- Most critical step: understanding the UoD.
- Goal: isolate relevant information to be represented in the IS.
  - Every relevant piece of information: must be elementary or derivable.
  - $\rightarrow$ Isolate each elementary fact.
    - Cannot be split into smaller units of information.
    - Simple assertion, atomic proposition about the UoD.
    - Epistemic commitment: people act as they *believed* the fact to be true.
- Questions: what kinds of info do we want from the system? Are entities well-identified? Can the facts be split into smaller units without losing information?
- Answers: by talking with domain experts about examples ("familiar information examples").
  - Reports, input forms, sample queries, ...
- Data use cases: talk about processes and requirements, but to understand the data. *Then* design the processes.

Elementary Fact

Asserts that a particular *object* has a property, or that one or more objects participate together in a relationship (each playing certain *role*).

- Ann smokes.
- Ann employs Bob.
- Bob is employed by Ann.
- If Ann employs Bob, then Bob gets a salary.
- If someone becomes employed, then he/she gets a salary.
- Lee is located in E301.
- Ann employs Bob and John.
- Ann and Bob open a loan request.
- Bob does not smoke. (disambiguate)

Elementary Fact

Asserts that a particular *object* has a property, or that one or more objects participate together in a relationship (each playing certain *role*).

- Ann smokes.
- Ann employs Bob.
- Bob is employed by Ann.
- If Ann employs Bob, then Bob gets a salary.
- If someone becomes employed, then he/she gets a salary.
- Lee is located in E301.
- Ann employs Bob and John. (!!!)
- Ann and Bob open a loan request. (disambiguate)
- Bob does not smoke. (disambiguate)
  - CWA vs OWA (with consistency constraint $A \wedge \neg A \rightarrow \bot$).
  - What about "Bob is a non-smoker"?

Basic Objects

- Value: has self-identifying reference (30, π, 'Lee', 'E301').
  - Rigid.
  - Strings and numbers.
- Entity/Object: referenced by a *definite description* (Lee, E301).
  - Typically changes with time.
  - Tangible (this computer) vs abstract (this lesson).
  - Referenced by a *rigid* value: use/mention distinction.
    - Lee is located in E301 vs 'Lee' is located in 'E301'.
  - Just a value is not sufficient $\rightarrow$ referential ambiguity.

What is a Definite Description?

Definite description

1. value ('Lee')
2. + explicit entity type (the `Person` 'Lee')...
3. + reference mode: the manner in which the value refers to the entity type (the `Person` with surname 'Lee').

Compact verbalization:
Person (.surname) 'Lee' is located in Room (.code) 'E301'.

Notes:

- Also composite identification schemes exist (later...).
- In critical cases, add a descriptive comment.

Roles

Modeled by logical predicates: sentences containing "object holes".

- Object hole: placeholder for an object designator (object term).
  The person with firstname 'Ann' *smokes* $\rightarrow$ ... *smokes* (unary).
- Most predicates: binary.
  The person with firstname 'Ann' *employs* the person with firstname 'Bob' $\rightarrow$ ... *employs* ...
- Extension to arbitrary n-ary predicates.
- Principles:
  - Order matters.
  - The n object terms must not be necessarily distinct.
  - The obtained proposition must not be expressible as a conjunction of simpler independent propositions.

Procedure

1. Collect significant reports, incomplete sentences, tables, graphs.
   - Cover all the possible cases.
   - Remember: most material represents *incomplete* knowledge.
2. Analyze them with domain expert using the telephone heuristic.
   - Identify synonyms, choose preferred terms, write a glossary.

$\rightarrow$ verbalized information about the system as-is.

Procedure

1. Collect significant reports, incomplete sentences, tables, graphs.
   - Cover all the possible cases.
   - Remember: most material represents *incomplete* knowledge.
2. Analyze them with domain expert using the telephone heuristic.
   - Identify synonyms, choose preferred terms, write a glossary.

   $\rightarrow$ verbalized information about the system as-is.
3. Process the verbalized information (modeler). Questions: which aspects should be modeled? Which parts may take on different values?
   - Write further examples.
   - Identify hidden constraints.
     - ★ Example: consider $A \wedge B \wedge C$.
       B and C independent $\rightarrow A \wedge B$; $A \wedge C$.
   - Rewrite information using definite descriptions for entities and identifying inverse roles.

   $\rightarrow$ elementary facts about the system as-is.

Procedure

1. Collect significant reports, incomplete sentences, tables, graphs.
   - Cover all the possible cases.
   - Remember: most material represents *incomplete* knowledge.
2. Analyze them with domain expert using the telephone heuristic.
   - Identify synonyms, choose preferred terms, write a glossary.

   $\rightarrow$ verbalized information about the system as-is.
3. Process the verbalized information (modeler). Questions: which aspects should be modeled? Which parts may take on different values?
   - Write further examples.
   - Identify hidden constraints.
     - ★ Example: consider $A \wedge B \wedge C$.
       B and C independent $\rightarrow A \wedge B$; $A \wedge C$.
   - Rewrite information using definite descriptions for entities and identifying inverse roles.

   $\rightarrow$ elementary facts about the system as-is.
4. Do the same with the new data requirements.

   $\rightarrow$ elementary facts about the system to-be.

Example

Tute Group	Time	Room	Student Nr	Student Name
A	Mon. 3 p.m.	CS-718	302156	Bloggs FB
			180064	Fletcher JB
			278155	Jackson M
B1	Tue. 2 p.m.	E-B18	266010	Anderson AB
			348112	Bloggs FB
...	...	...	...	...

Example

Tute Group	Time	Room	Student Nr	Student Name
A	Mon. 3 p.m.	CS-718	302156 180064 278155	Bloggs FB Fletcher JB Jackson M
B1	Tue. 2 p.m.	E-B18	266010 348112	Anderson AB Bloggs FB
...	...	...	...	...

Typical verbalization by domain expert:

- Student 302156 belongs to group A and is named 'Bloggs FB'.
- Tute group A meets at 3 p.m. Monday in Room CS-718.

Value Types, Inverse Roles

Student 302156 belongs to group A and is named 'Bloggs FB'.

- Name and surname together.
- Student name and nr. in the same row refer to the same student.
- Student has only one number but could share the name with others.
  - Student number is a good identifier, student name is not.

Value Types, Inverse Roles

Student 302156 belongs to group A and is named 'Bloggs FB'.

- Name and surname together.
- Student name and nr. in the same row refer to the same student.
- Student has only one number but could share the name with others.
  - Student number is a good identifier, student name is not.

$\rightarrow$ Student (nr.) 302156 *has* StudentName 'Bloggs FB'.

- StudentName is a value type: no reference scheme.

$\rightarrow$ Student (nr.) 302156 *belongs to* Tutegroup (.code) 'A'.

- Inverse: Tutegroup (.code) 'A' *involves* Student (nr.) 302156.
- $\ldots_{(Stud.)}$ belongs to $\ldots_{(TuteG.)} \leftrightarrow \ldots_{(TuteG.)}$ involves $\ldots_{(Stud.)}$
  - $\neq$ surface structure, $=$ deep structure.
  - One primary (mandatory), the inverse optional.

$\rightarrow$ Student (nr.) 302156 *belongs to/involves* Tutegroup (.code) 'A'.

(In)separability of Facts

Tute group A meets at 3 p.m. Monday in Room CS-718.

$\downarrow$

TuteGroup(.code) ‘A’ *meets at* Time(.dhcode) ‘Mon. 3 p.m.’ *in* Room(.code) ‘CS-718’.

- Hp: TuteGroups meet more than once a week.
  - Further questions (Always in the same room? Suppose not)
  - The fact is inseparable.
  - Hence elementary $\rightarrow$ a ternary predicate!
  - Need to complete the sample data with additional significant cases:
    - TuteGroup(.code) ‘A’ *meets at* Time(.dhcode) ‘Tue. 4 p.m.’ *in* Room(.code) ‘CS-513’.
  - Separation $\rightarrow$ information loss!

(In)separability of Facts

Tute group A meets at 3 p.m. Monday in Room CS-718.
↓
TuteGroup(.code) ‘A’ *meets at* Time(.dhcode) ‘Mon. 3 p.m.’ *in* Room(.code) ‘CS-718’.

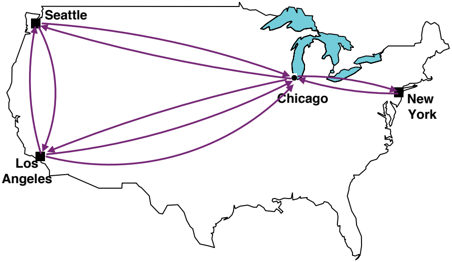
- Sample questions:
  1. Does TuteGroup(.code) ‘A’ always meet in Room(.code) ‘CS-718’?
  2. Does this hold for all TuteGroups?
  3. Do TuteGroups meet only once a week? (Note: $(3) \rightarrow (2)$).

(In)separability of Facts

Tute group A meets at 3 p.m. Monday in Room CS-718.
$\downarrow$
TuteGroup(.code) 'A' *meets at* Time(.dhcode) 'Mon. 3 p.m.' *in* Room(.code) 'CS-718'.

- Hp: TuteGroups meet only once a week.
  - The fact must be separated.
  - It is not elementary $\rightarrow$ two binary predicates!
  - TuteGroup(.code) 'A' *meets at* Time(.dhcode) 'Mon. 3 p.m.'.
    TuteGroup(.code) 'A' *meets in/hosts* Room(.code) 'CS-718'.

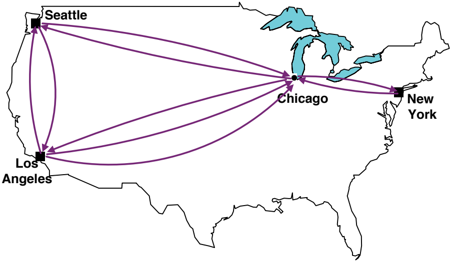
System As-Is vs System To-Be



System as-is: direct flight connections between cities.

- City(.name) 'New York' *has a flight to*/*has a flight from* City(.name) 'Chicago'.

System As-Is vs System To-Be



System as-is: direct flight connections between cities.

- City(.name) ‘New York’ *has a flight to/has a flight from* City(.name) ‘Chicago’.

System to-be:

- Info about the flights.
- Notion of airport.
- Notion of airport that serves one or more cities.