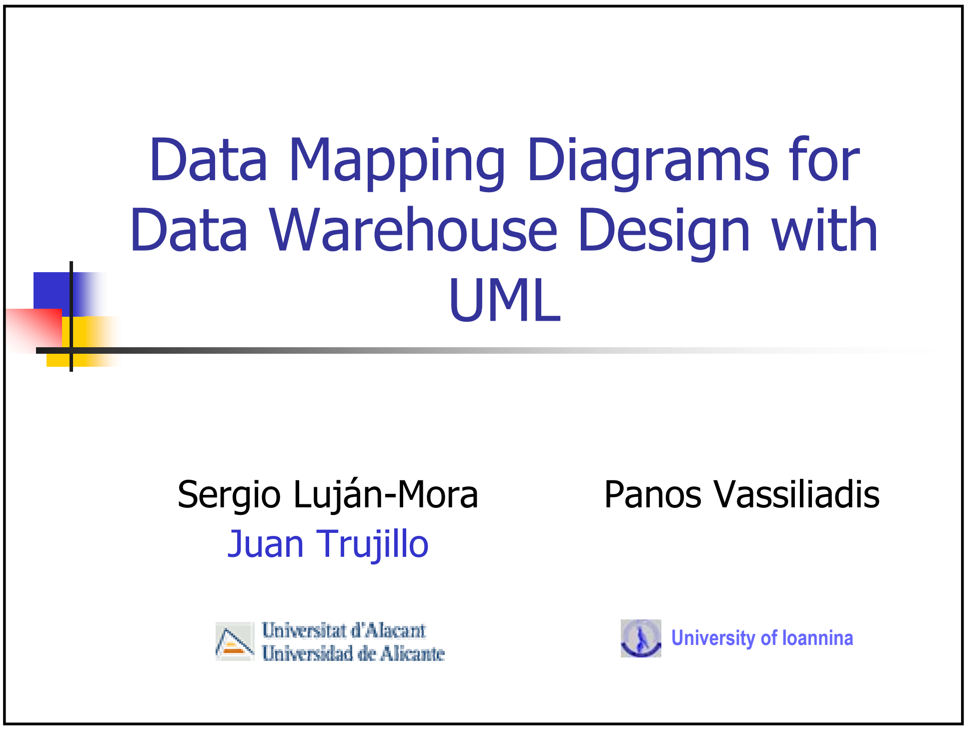
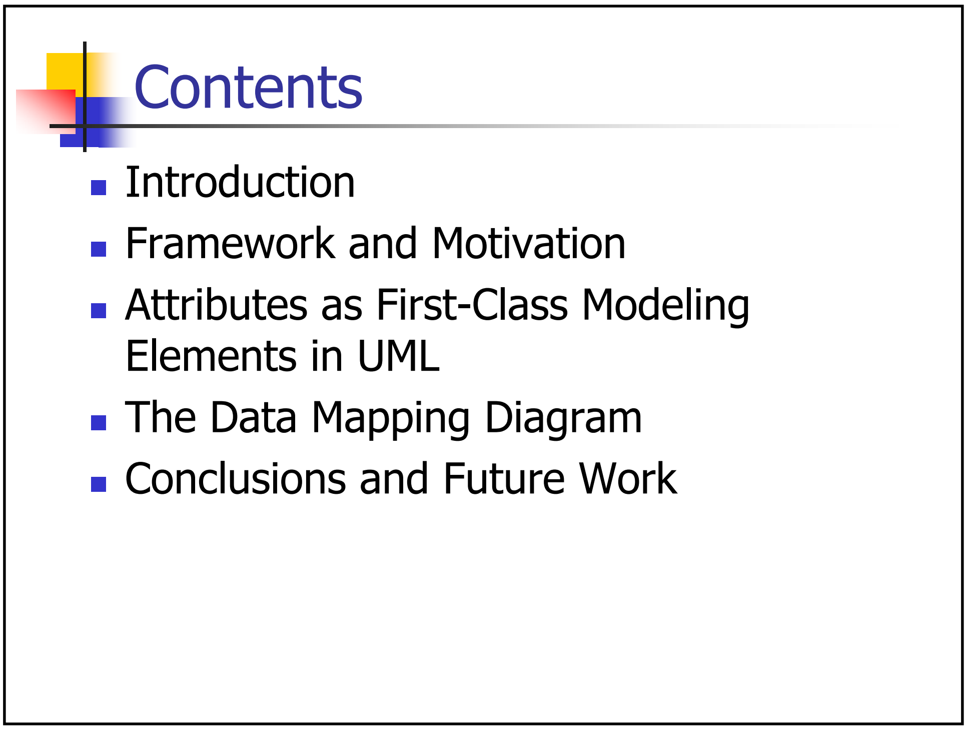
23rd International Conference on Conceptual Modeling (ER 2004), p. 191-204: Lecture Notes in Computer Science 3288, Shanghai (China), November 8-12 2004.



Data Mapping Diagrams for Data Warehouse Design with UML

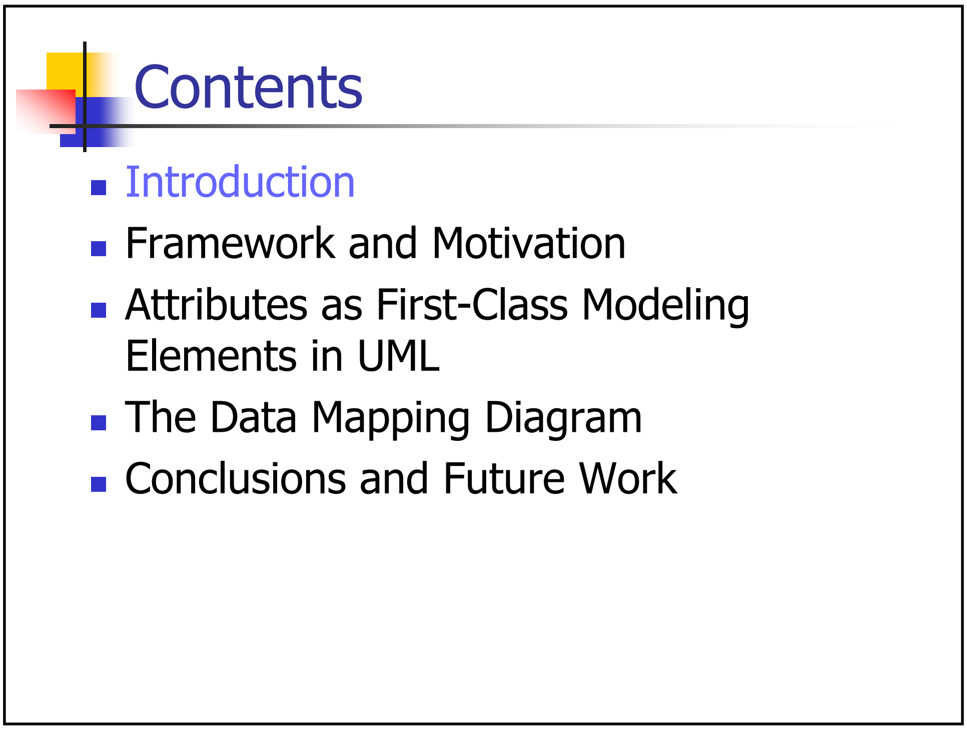
Sergio Luján-Mora
Juan Trujillo
Universitat d'Alacant
Universidad de Alicante

Panos Vassiliadis
University of Ioannina



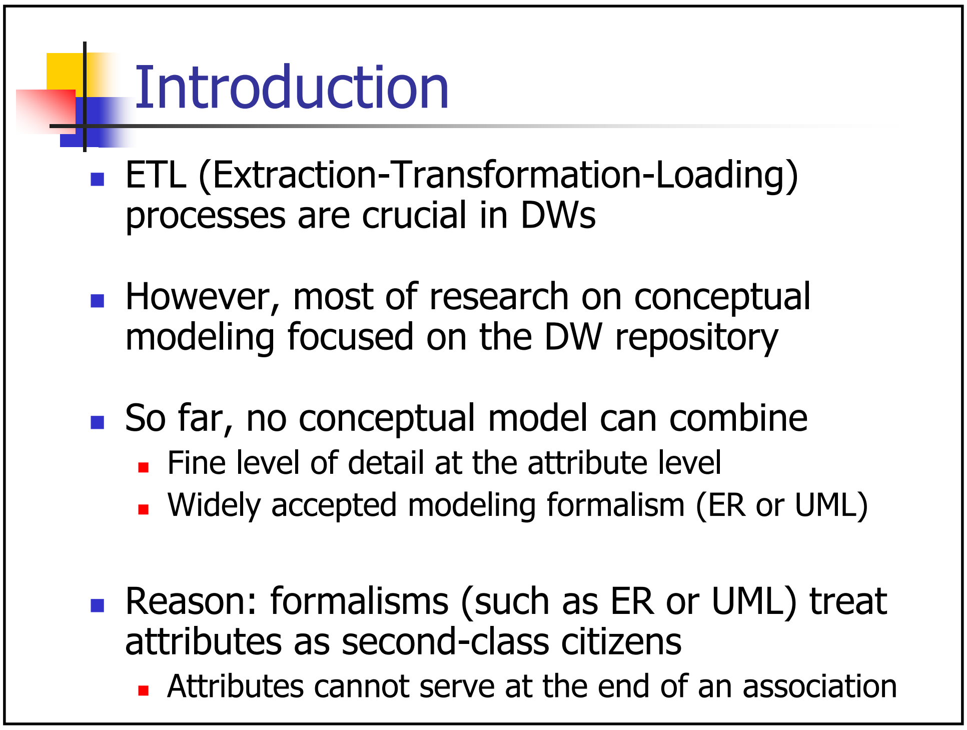
Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



Introduction

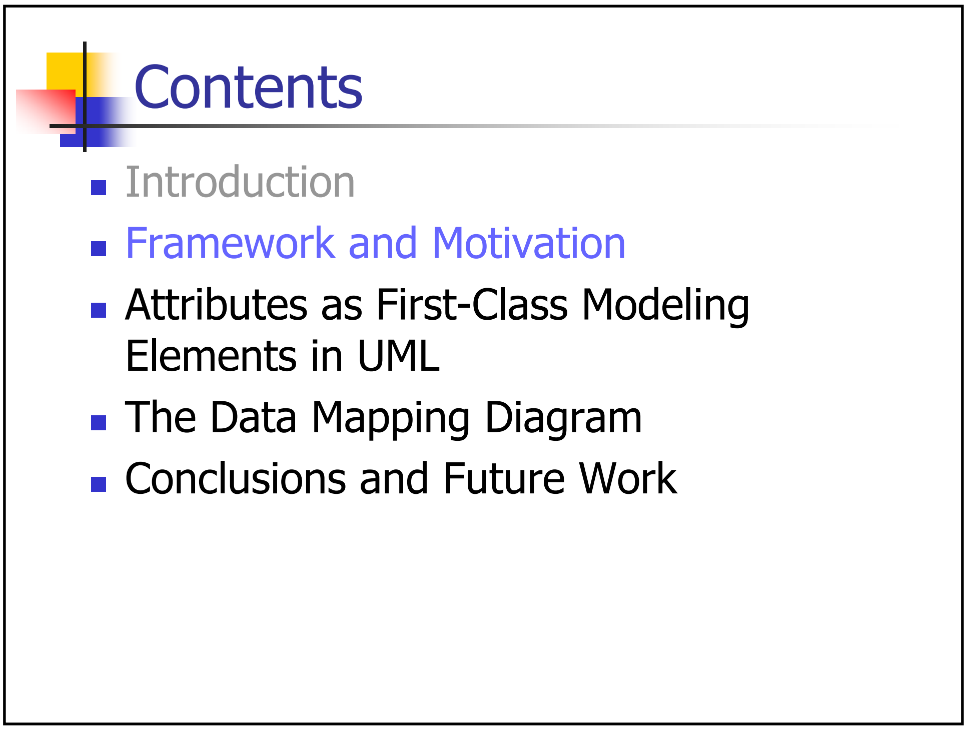
- ETL (Extraction-Transformation-Loading) processes are crucial in DWs
- However, most of research on conceptual modeling focused on the DW repository
- So far, no conceptual model can combine
  - Fine level of detail at the attribute level
  - Widely accepted modeling formalism (ER or UML)
- Reason: formalisms (such as ER or UML) treat attributes as second-class citizens
  - Attributes cannot serve at the end of an association

Introduction

- Our proposal: a new UML diagram called the "*data mapping diagram*" to model ETL processes at various levels of detail
  - Attributes can be treated as first-class citizens (*FCML*)
  - UML is extended through the extension mechanisms
  - Attributes are formally mapped to proxy classes that can participate in associations for inter-attributes mappings
- KEY ASPECT
  - Integrated in a global approach for the modeling of DWs, based on the UML (ER02 & 03, UML02, IEEE)

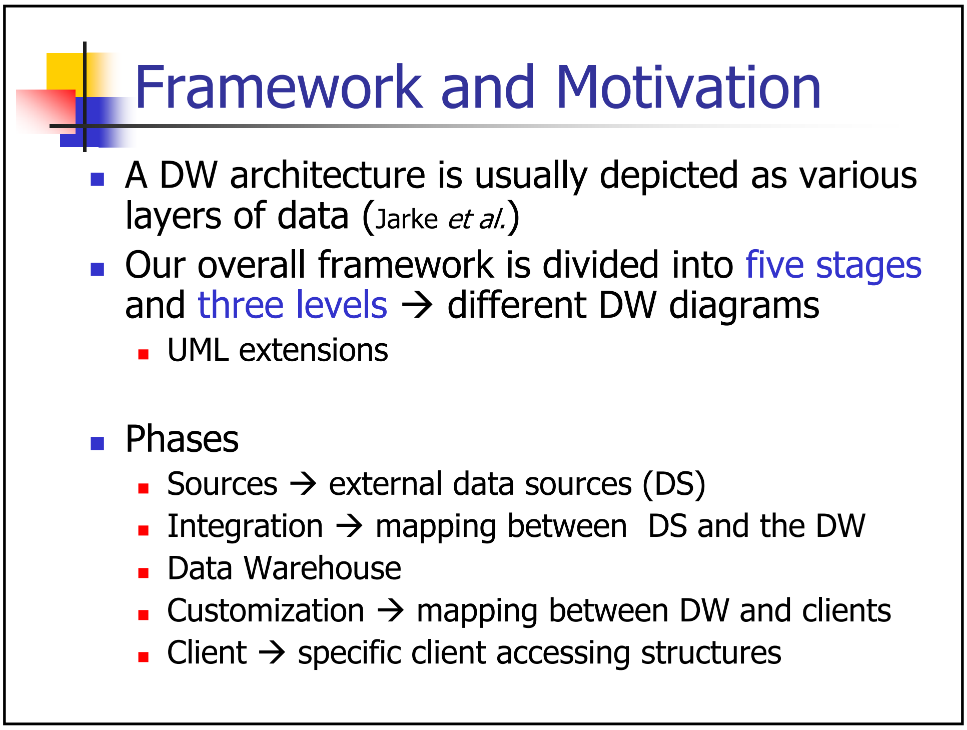
Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



Framework and Motivation

- A DW architecture is usually depicted as various layers of data (Jarke *et al.*)
- Our overall framework is divided into five stages and three levels → different DW diagrams
  - UML extensions
- Phases
  - Sources → external data sources (DS)
  - Integration → mapping between DS and the DW
  - Data Warehouse
  - Customization → mapping between DW and clients
  - Client → specific client accessing structures

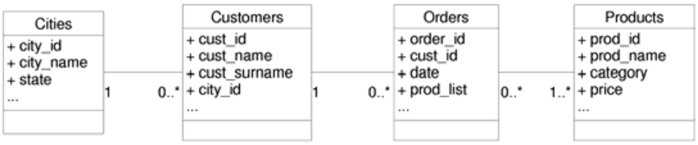
Framework and Motivation

- Levels
  - Conceptual → DW conceptual modeling
  - Logical → logical aspects of the DW (e.g. ETL)
  - Physical → e.g. storage of the ETL processes

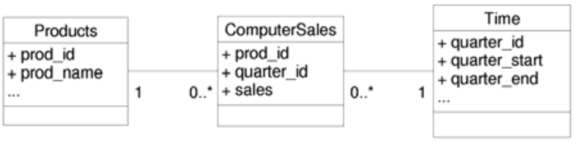
- Diagrams → Several UML profiles have been proposed. In this work: *Data mapping diagram*

- Motivating Example
  - Building a DW from the retail system of a company

Framework and Motivation

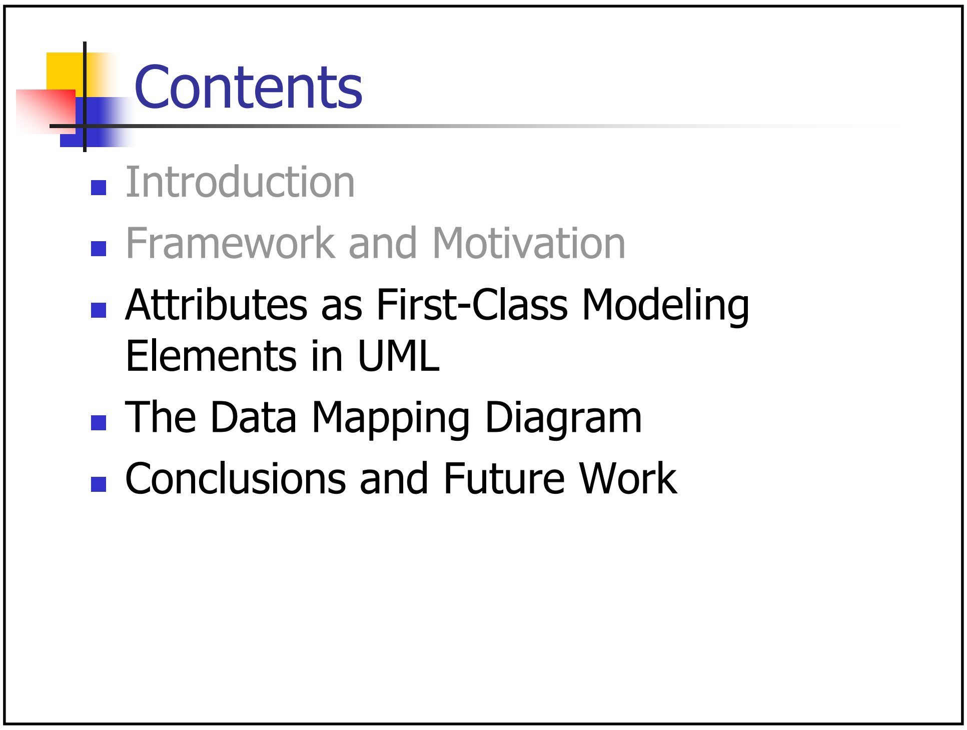


Source Conceptual Schema (SCS)



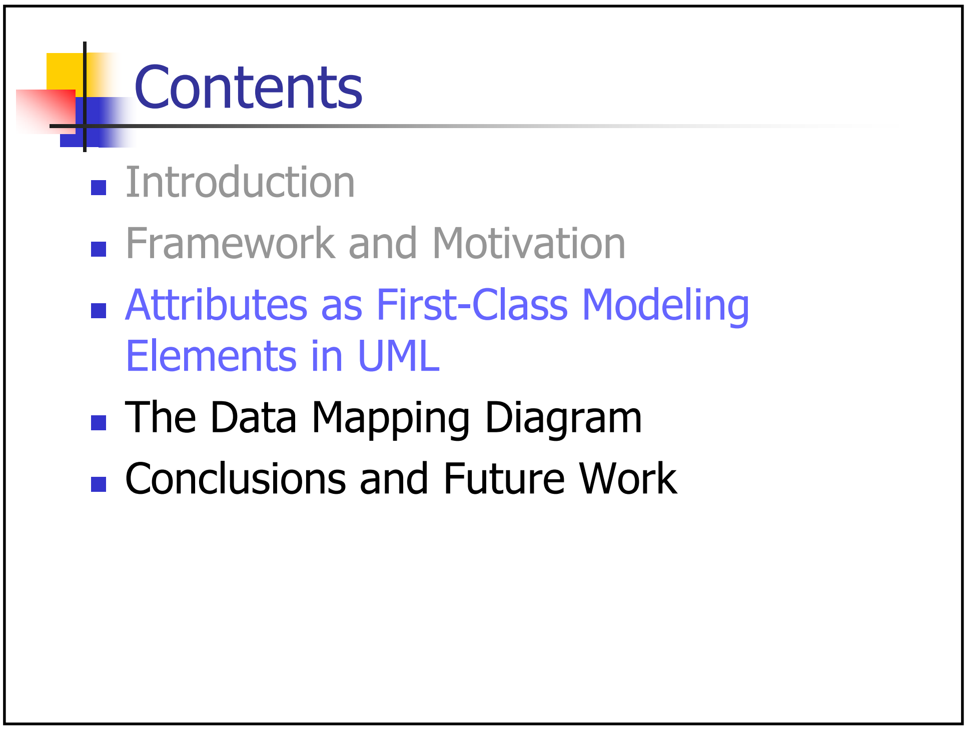
Data Warehouse Conceptual Schema (DWCS)

- Data Mapping Diagram shows the relationships between the data sources and the DW, and between the DW and client's structures



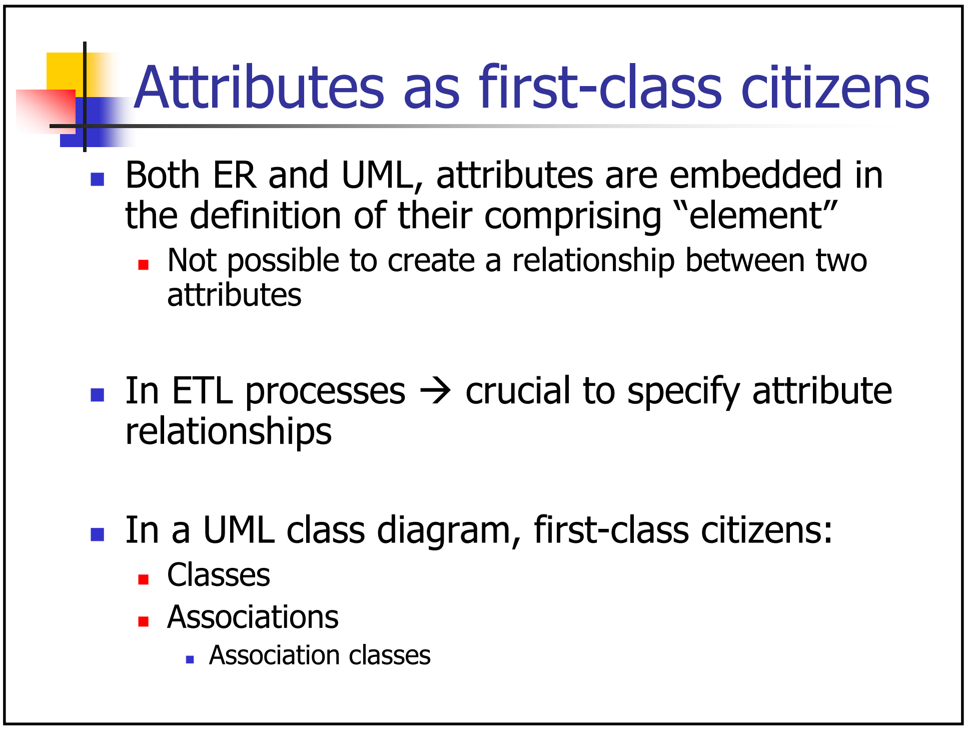
Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



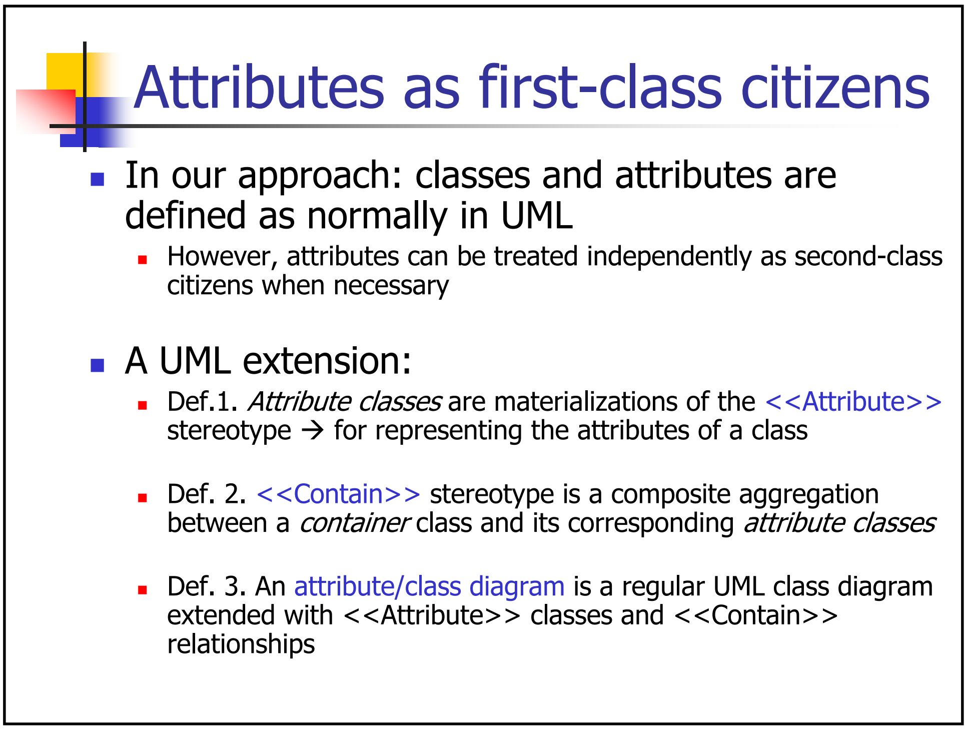
Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



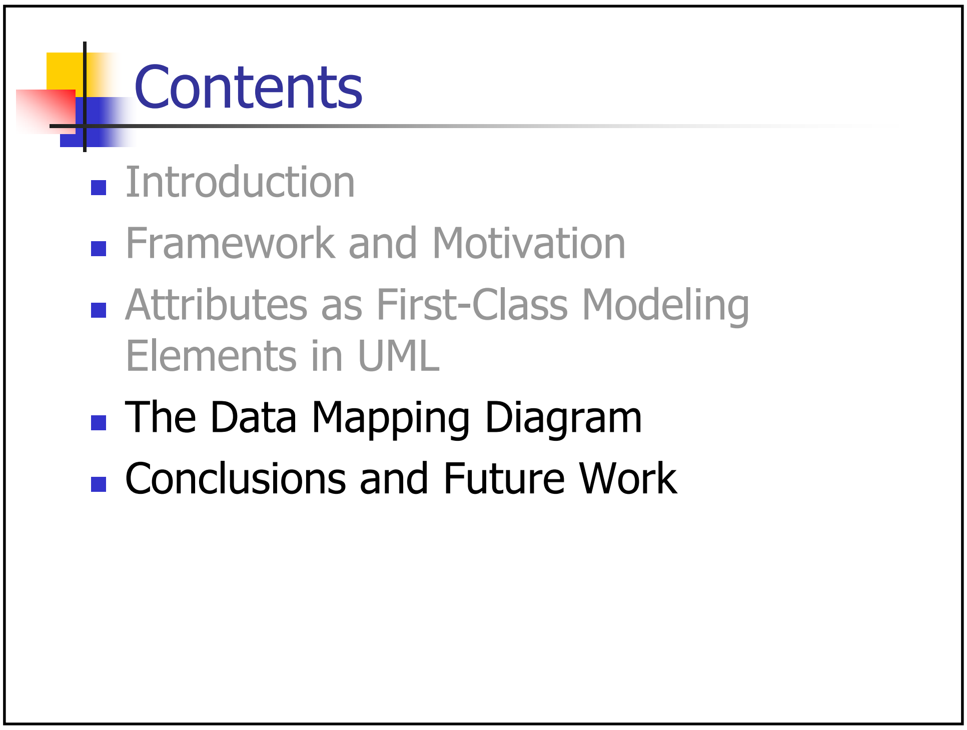
Attributes as first-class citizens

- Both ER and UML, attributes are embedded in the definition of their comprising "element"
  - Not possible to create a relationship between two attributes
- In ETL processes → crucial to specify attribute relationships
- In a UML class diagram, first-class citizens:
  - Classes
  - Associations
    - Association classes



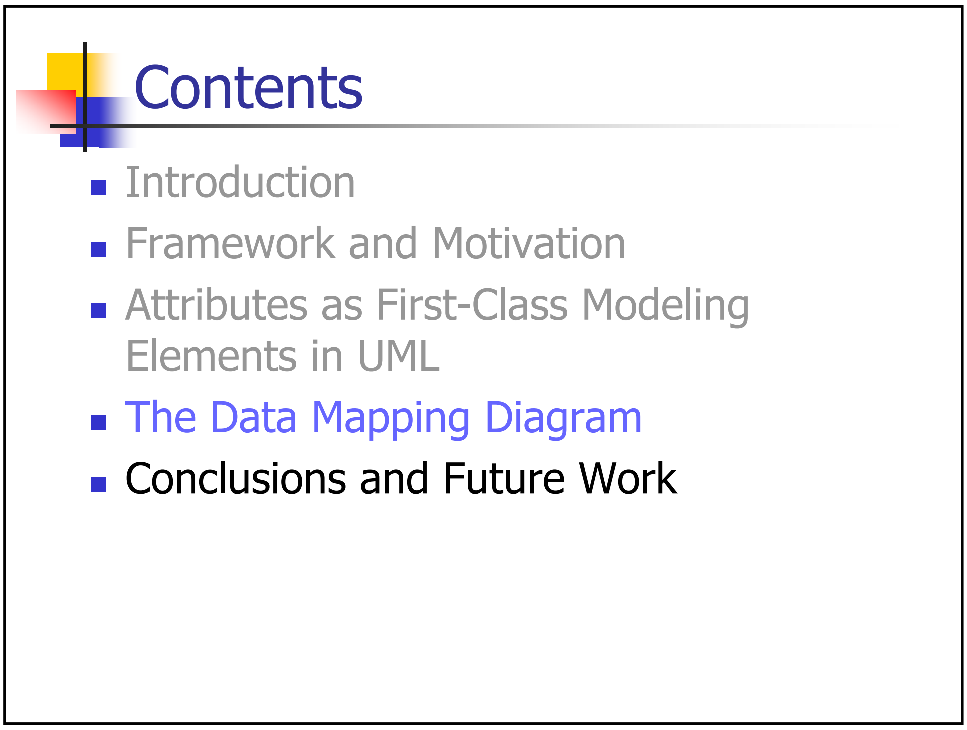
Attributes as first-class citizens

- In our approach: classes and attributes are defined as normally in UML
  - However, attributes can be treated independently as second-class citizens when necessary
- A UML extension:
  - Def.1. *Attribute classes* are materializations of the <<Attribute>> stereotype → for representing the attributes of a class
  - Def. 2. <<Contain>> stereotype is a composite aggregation between a *container* class and its corresponding *attribute classes*
  - Def. 3. An attribute/class diagram is a regular UML class diagram extended with <<Attribute>> classes and <<Contain>> relationships



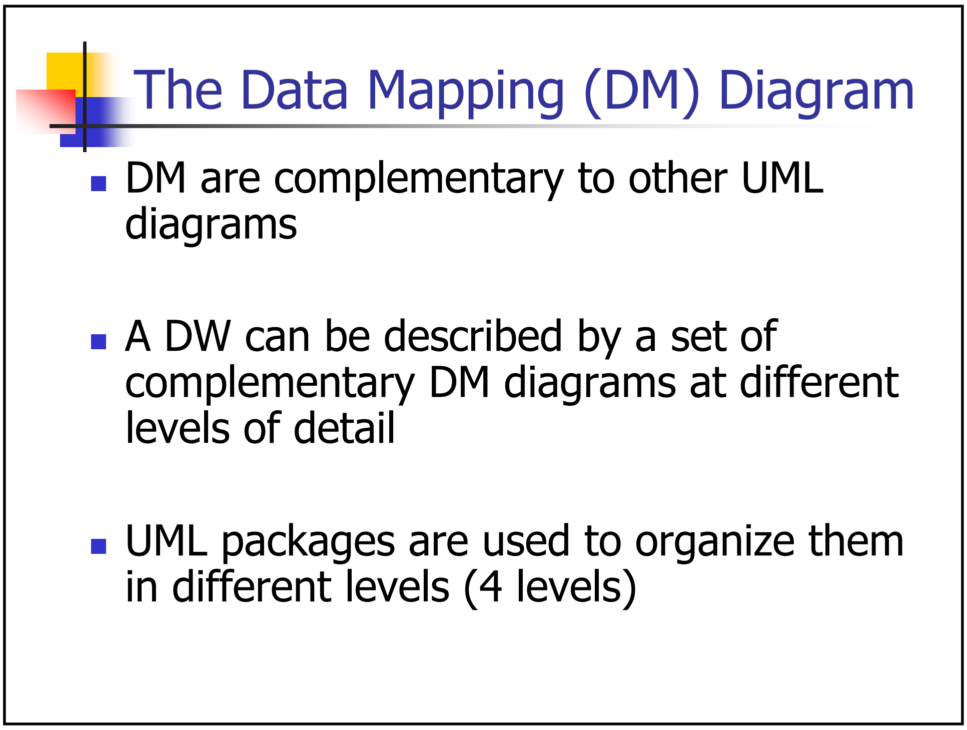
Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



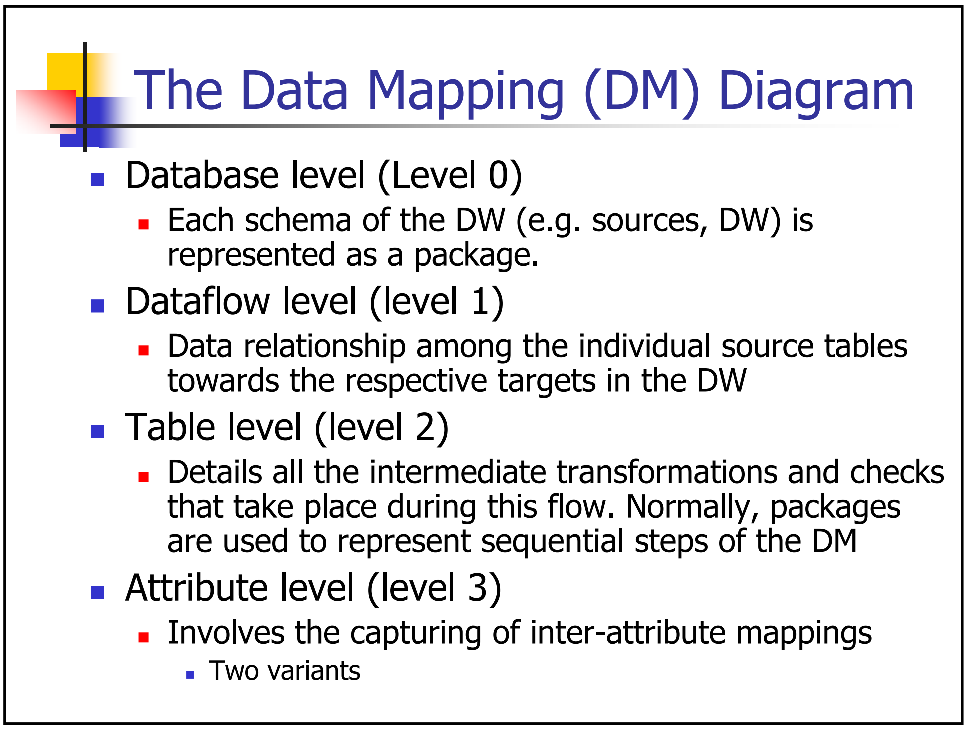
Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



The Data Mapping (DM) Diagram

- DM are complementary to other UML diagrams
- A DW can be described by a set of complementary DM diagrams at different levels of detail
- UML packages are used to organize them in different levels (4 levels)

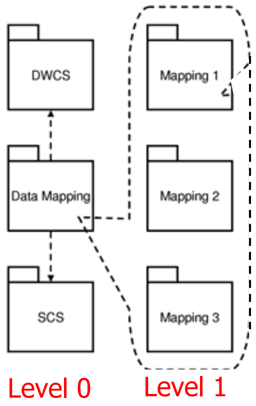


The Data Mapping (DM) Diagram

- Database level (Level 0)
  - Each schema of the DW (e.g. sources, DW) is represented as a package.
- Dataflow level (level 1)
  - Data relationship among the individual source tables towards the respective targets in the DW
- Table level (level 2)
  - Details all the intermediate transformations and checks that take place during this flow. Normally, packages are used to represent sequential steps of the DM
- Attribute level (level 3)
  - Involves the capturing of inter-attribute mappings
    - Two variants

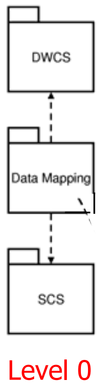
The Data Mapping Diagram

- Database and dataflow diagrams (0 and 1) use traditional UML elements
  - Packages for modeling data relationships
  - Dependency relationships between packages



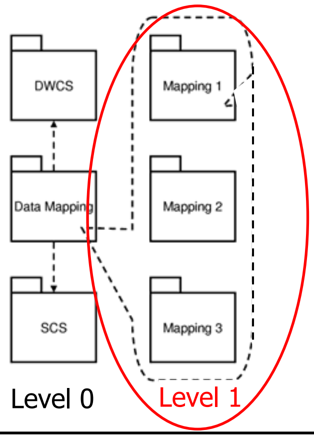
The Data Mapping Diagram

- Level 0: a simple relationship among the DWCS and the SCS exists
  - Captured by a single Data Mapping package and these three design elements



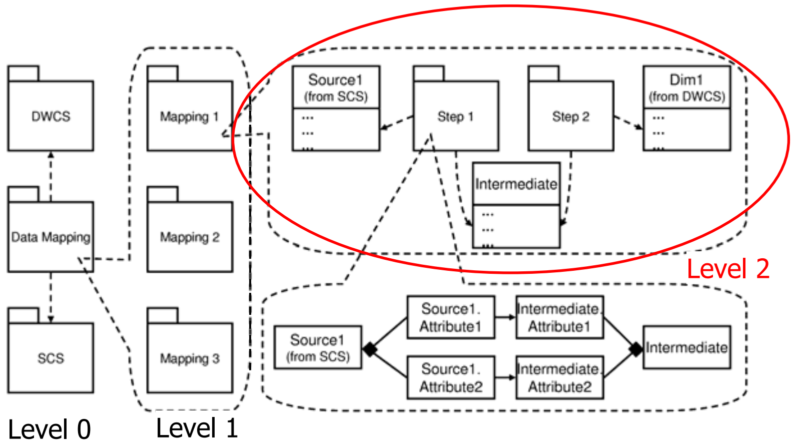
The Data Mapping Diagram

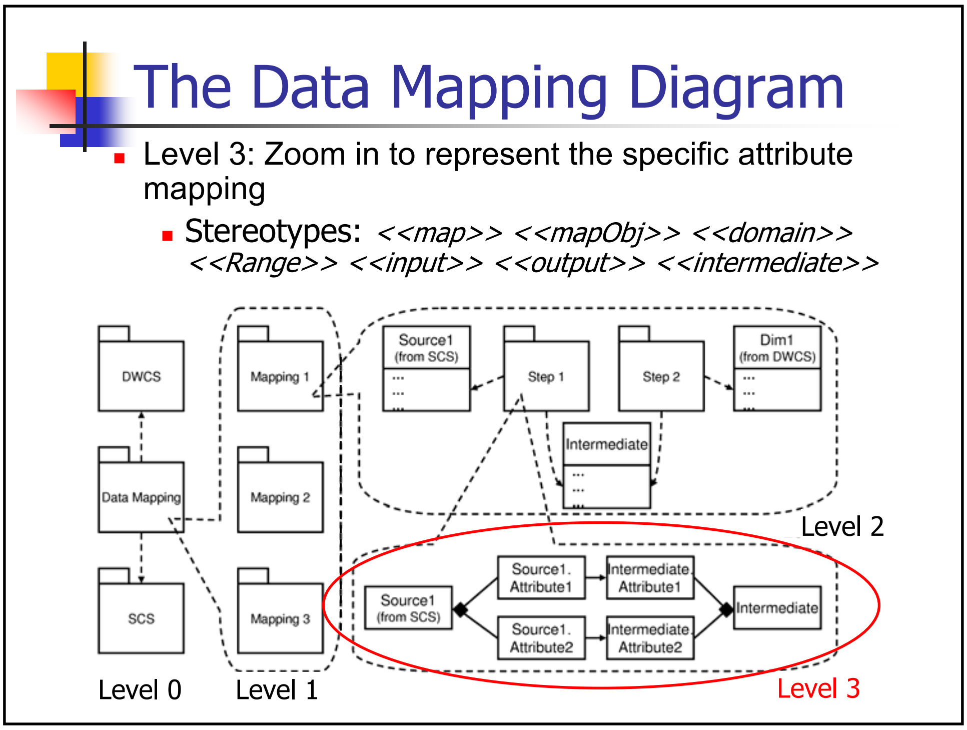
- Level 1: three particular tables to populate → three main scenarios for populating the DW
  - Captured by three Data Mapping packages



The Data Mapping Diagram

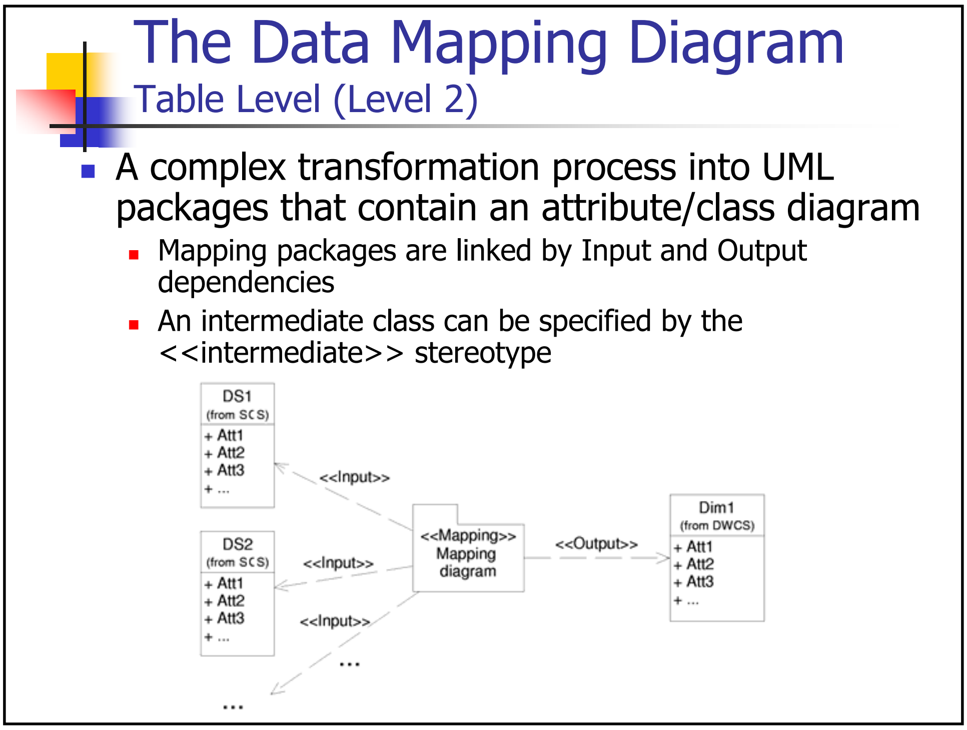
- Level 2: Zoom in to represent the particular transformation/mapping
  - Stereotypes: *<<mapping>> <<input>> <<output>>*





The Data Mapping Diagram

- Level 3: Zoom in to represent the specific attribute mapping
  - Stereotypes: *<<map>> <<mapObj>> <<domain>> <<Range>> <<input>> <<output>> <<intermediate>>*



The Data Mapping Diagram

Table Level (Level 2)

- A complex transformation process into UML packages that contain an attribute/class diagram
  - Mapping packages are linked by Input and Output dependencies
  - An intermediate class can be specified by the <<intermediate>> stereotype

The Data Mapping Diagram

Attribute Level (Level 3)

- Relationships between the attributes of the classes involved in a data mapping. Two variants:
  - Compact Variant: relationships represented as *associations*, and the semantic of the mapping described in a UML *note*
    - Less cluttered diagrams, but less semantic impact
  - Formal variant: relationships represented by a *mapping object*, and the semantic of the mapping described in a *tag definition*
    - More modeling elements, but more formal semantics

The Data Mapping Diagram

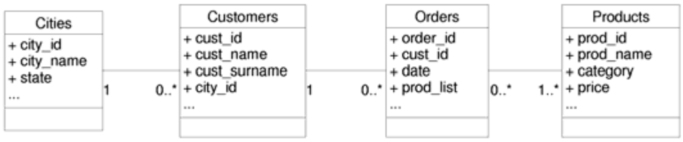
Attribute Level (Level 3). Compact variant

- Elements are imported from other diagrams
- Attributes are represented as <<Attribute>> classes
- <<Attribute>> classes connected by association relationships and the navigability
- Association relationships adorned with the stereotype <<Map>>
- UML notes to specify how the target attribute of the DW is obtained from the source one

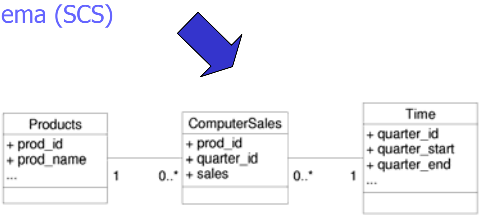
The Data Mapping Diagram

Motivating example

- Calculate the quarterly sales of the products belonging to the computer category



Source Conceptual Schema (SCS)

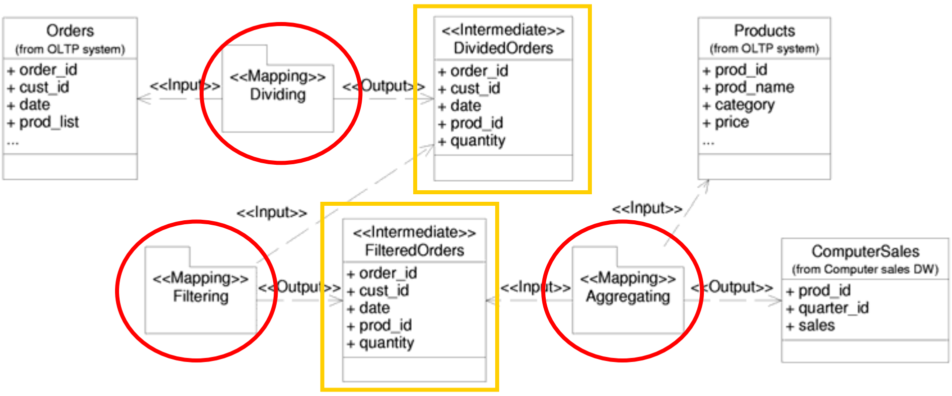


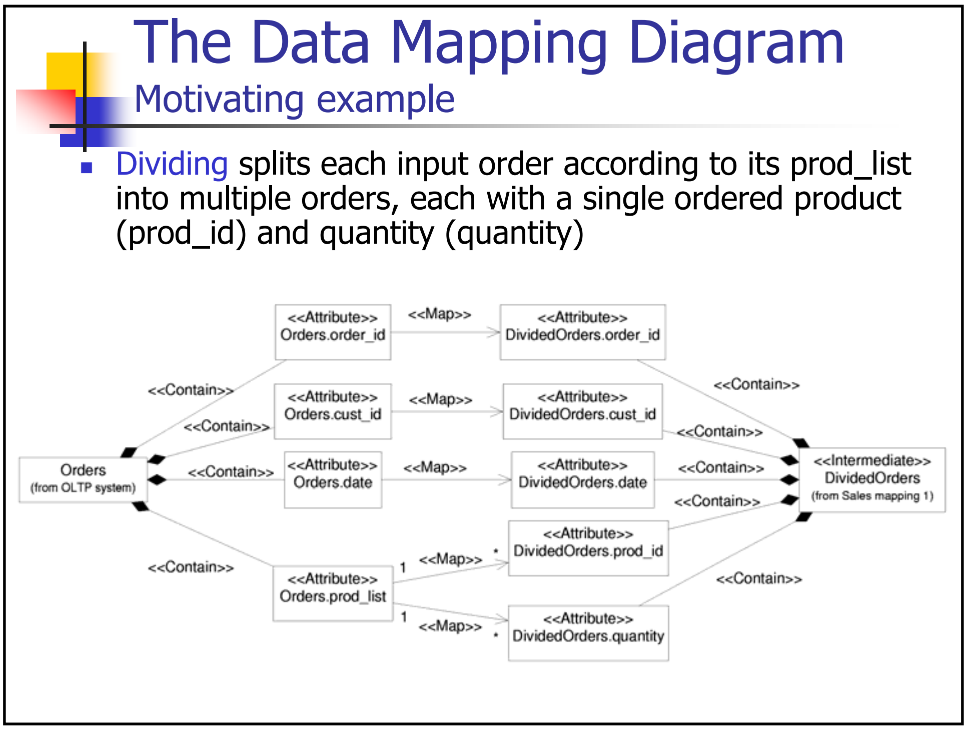
Data Warehouse Conceptual Schema (DWCS)

The Data Mapping Diagram

Motivating example

- Level 2 of the data mapping diagram



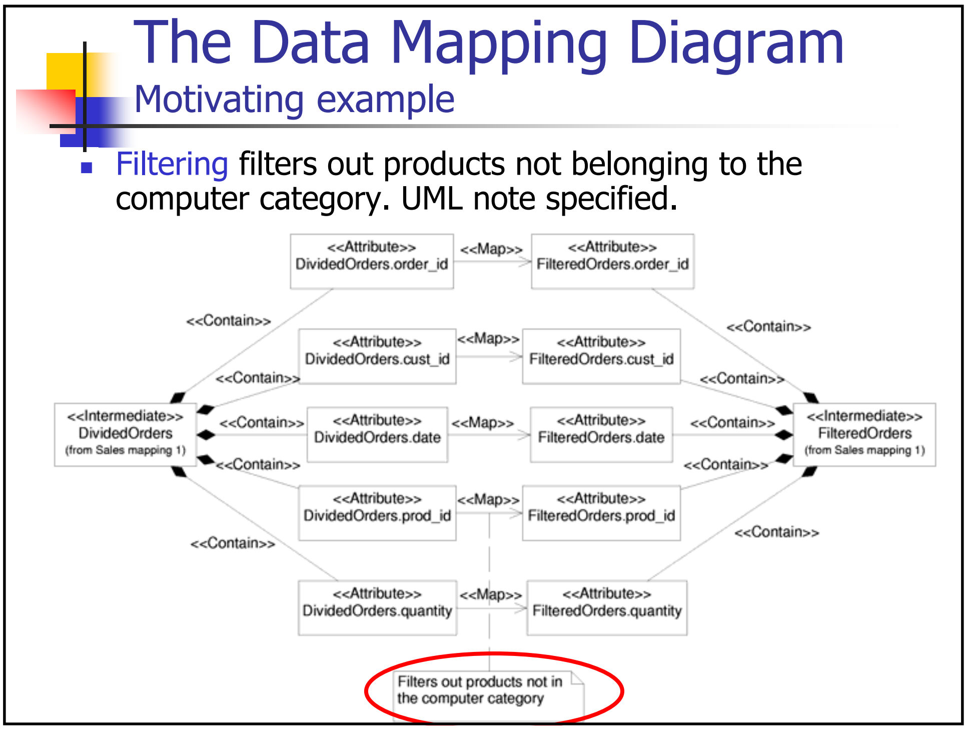


The Data Mapping Diagram

Motivating example

- Dividing splits each input order according to its prod_list into multiple orders, each with a single ordered product (prod_id) and quantity (quantity)

<<Attribute>> Orders.order_id
<<Map>>
<<Attribute>> DividedOrders.order_id
<<Contain>>
<<Attribute>> Orders.cust_id
<<Map>>
<<Attribute>> DividedOrders.cust_id
<<Contain>>
<<Contain>>
<<Contain>>
Orders (from OLTP system)
<<Contain>>
<<Attribute>> Orders.date
<<Map>>
<<Attribute>> DividedOrders.date
<<Contain>>
<<Intermediate>> DividedOrders (from Sales mapping 1)
<<Contain>>
<<Attribute>> DividedOrders.prod_id
<<Contain>>
1
<<Map>>
*
<<Attribute>> Orders.prod_list
<<Contain>>
1
<<Map>>
*
<<Attribute>> DividedOrders.quantity

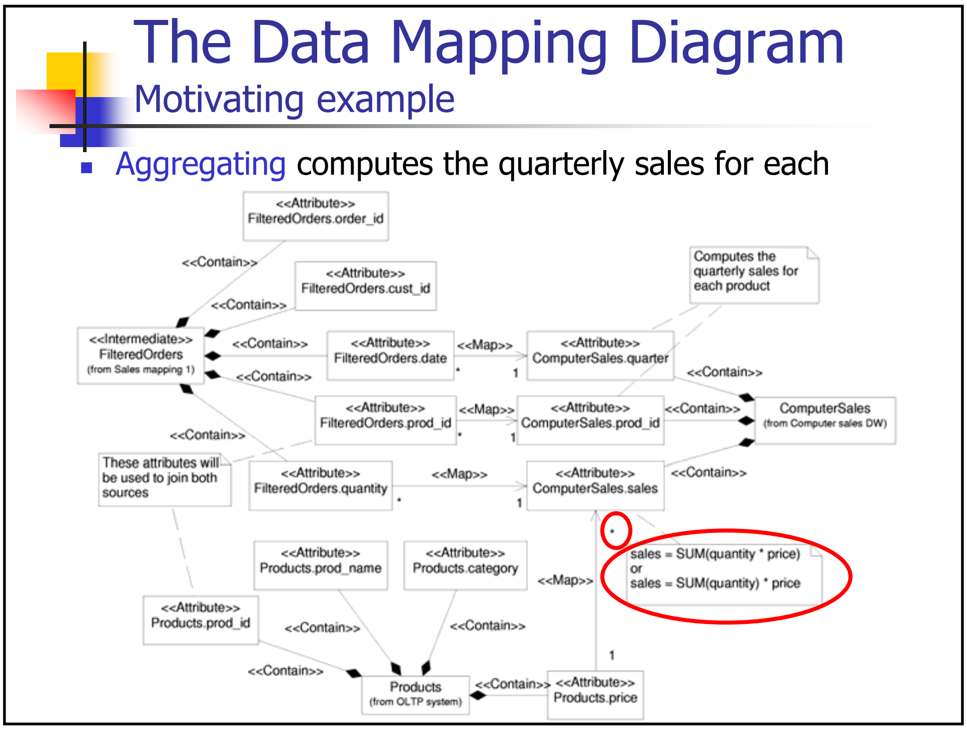


The Data Mapping Diagram

Motivating example

- Filtering filters out products not belonging to the computer category. UML note specified.

<<Attribute>> DividedOrders.order_id
<<Map>>
<<Attribute>> FilteredOrders.order_id
<<Contain>>
<<Contain>>
<<Attribute>> DividedOrders.cust_id
<<Map>>
<<Attribute>> FilteredOrders.cust_id
<<Contain>>
<<Contain>>
<<Intermediate>> DividedOrders (from Sales mapping 1)
<<Contain>>
<<Attribute>> DividedOrders.date
<<Map>>
<<Attribute>> FilteredOrders.date
<<Contain>>
<<Intermediate>> FilteredOrders (from Sales mapping 1)
<<Contain>>
<<Contain>>
<<Attribute>> DividedOrders.prod_id
<<Map>>
<<Attribute>> FilteredOrders.prod_id
<<Contain>>
<<Contain>>
<<Attribute>> DividedOrders.quantity
<<Map>>
<<Attribute>> FilteredOrders.quantity
Filters out products not in the computer category

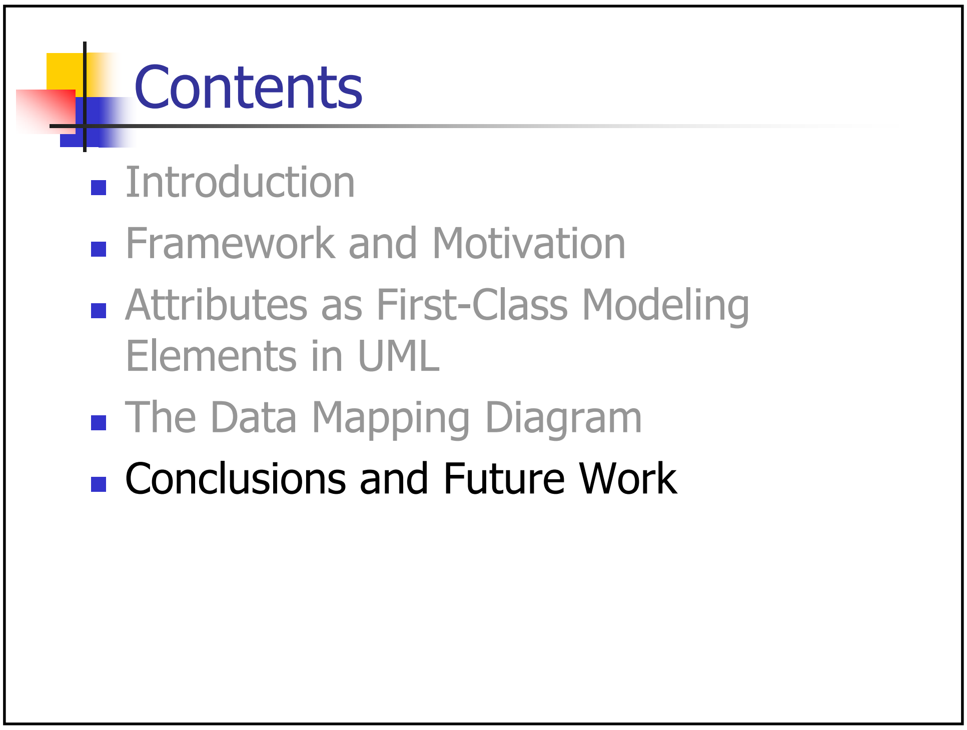


The Data Mapping Diagram

Motivating example

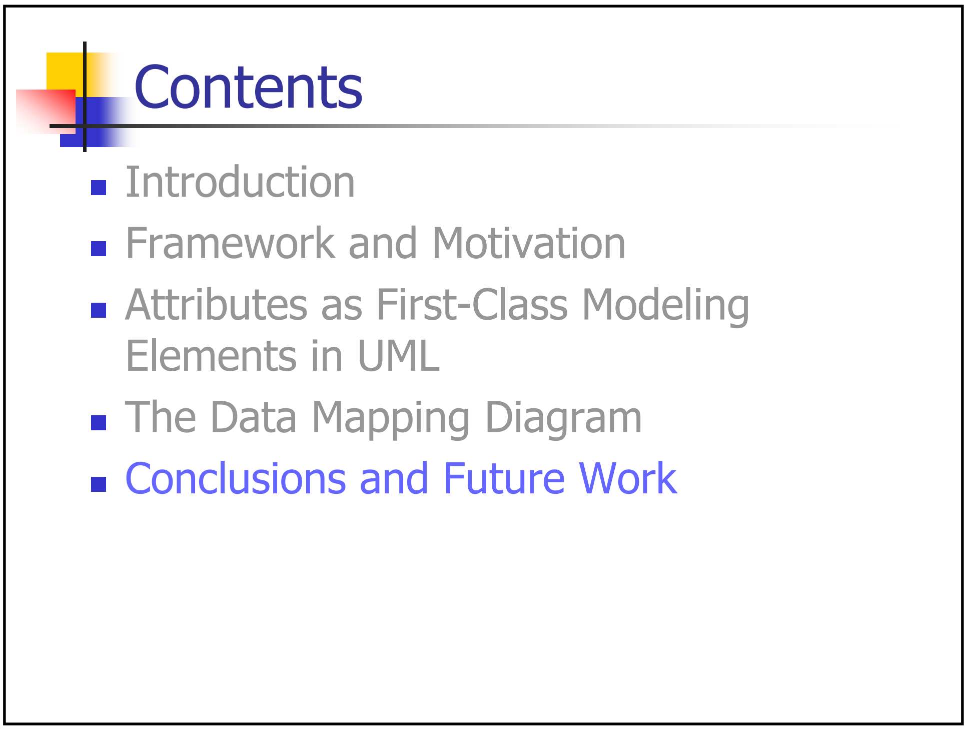
- Aggregating computes the quarterly sales for each

<<Attribute>> FilteredOrders.order_id
<<Contain>>
<<Attribute>> FilteredOrders.cust_id
<<Contain>>
<<Intermediate>> FilteredOrders (from Sales mapping 1)
<<Contain>>
<<Attribute>> FilteredOrders.date
<<Map>>
<<Attribute>> ComputerSales.quarter
Computes the quarterly sales for each product
<<Contain>>
<<Attribute>> FilteredOrders.prod_id
<<Map>>
<<Attribute>> ComputerSales.prod_id
<<Contain>>
ComputerSales (from Computer sales DW)
<<Contain>>
These attributes will be used to join both sources
<<Attribute>> FilteredOrders.quantity
<<Map>>
<<Attribute>> ComputerSales.sales
<<Contain>>
*
sales = SUM(quantity * price) or sales = SUM(quantity) * price
<<Attribute>> Products.prod_name
<<Attribute>> Products.category
<<Map>>
<<Attribute>> Products.prod_id
<<Contain>>
<<Contain>>
<<Contain>>
Products (from OLTP system)
<<Contain>>
<<Attribute>> Products.price
1



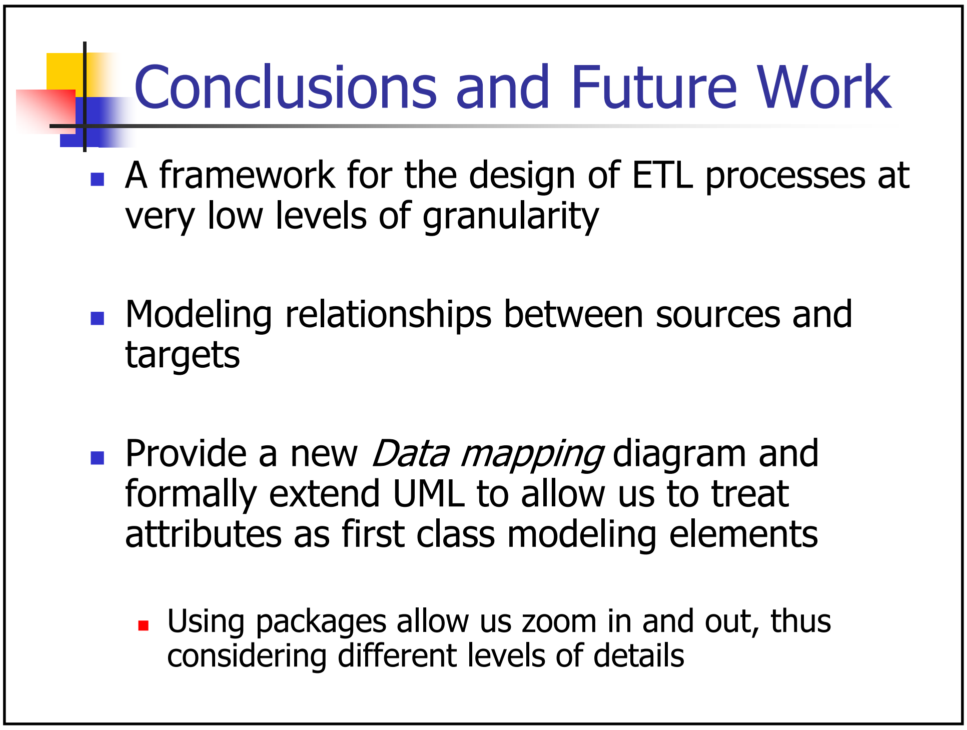
Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



Contents

- Introduction
- Framework and Motivation
- Attributes as First-Class Modeling Elements in UML
- The Data Mapping Diagram
- Conclusions and Future Work



Conclusions and Future Work

- A framework for the design of ETL processes at very low levels of granularity
- Modeling relationships between sources and targets
- Provide a new *Data mapping* diagram and formally extend UML to allow us to treat attributes as first class modeling elements
  - Using packages allow us zoom in and out, thus considering different levels of details

Conclusions and Future Work

- Immediate future work
  - Implementing the Compact approach
    - Considering more formal semantics instead of notes
  - A complete implementation mapping in a target platform and existing ETL tools (e.g. DB2 Center, OWB and so on)
- Further work
  - This approach can be applied to the modeling of XML documents, specification of Web services or materialization of database views.

Data Mapping Diagrams for Data Warehouse Design with UML

Sergio Luján-Mora
Juan Trujillo
Universitat d'Alacant
Universidad de Alicante

Panos Vassiliadis
University of Ioannina