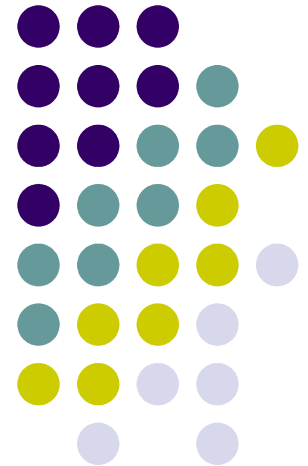
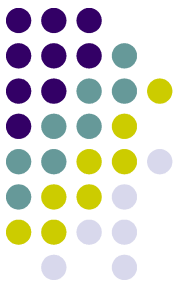


# Ontology Mapping Survey

---

Siyamed Seyhmus SINIR





# Introduction

## **Why do we use ontology?**

To describe the semantics of the data (which we name as Meta-Data)

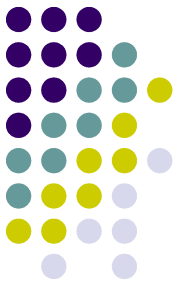
## **Why do we describe the semantics?**

In order to provide a uniform way to make different parties to understand each other

## **Which data?**

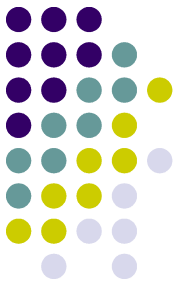
Any data (on the web, or in the existing legacy databases)

# Introduction



## Formal definition on Ontology:

Ontologies are knowledge bodies that provide a **formal representation** of a **shared conceptualization** of a particular domain.

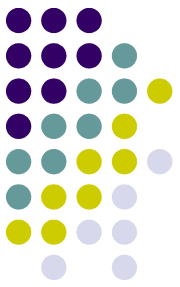


# Introduction

Ontologies are widely used in the Semantic Web.

Recently ontologies have become increasingly common on WWW where they provide semantics of annotations in web pages

This distributed nature of ontology development has led to a large number of **different** ontologies covering **the same** or **overlapping domains**.



# Introduction

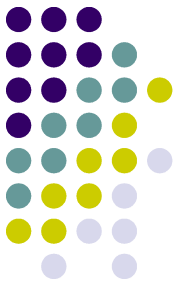
In order to two parties to understand each other, they should use the same **formal representation** for the **shared conceptualization** (so the same ontology)

Unfortunately it is not easy to make everybody to agree on the same ontology for a domain

And when you have different ontologies for the same domain the problem shows up.

Parties with different ontologies do not understand each other.

Here comes the ontology mapping into the play



# Introduction

This is not a new problem

Same thing exists in **Federated Databases** and other **Data Integration** efforts

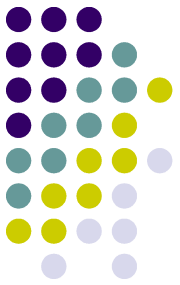
In federated databases there are **local schemas** of the individual databases or database groups.

Either provide bilateral mappings for each of the databases (which result in  $n^2$  mappings) or

Define a **global schema** to include all of the others which has a mapping to each of them

Generally mapping is done via **views**.

Global as view, Local as view



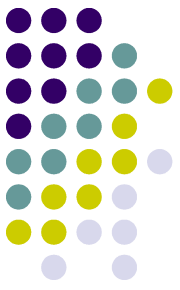
# Overview

## Ontology Mapping (OM)

Schema Matching (SM)

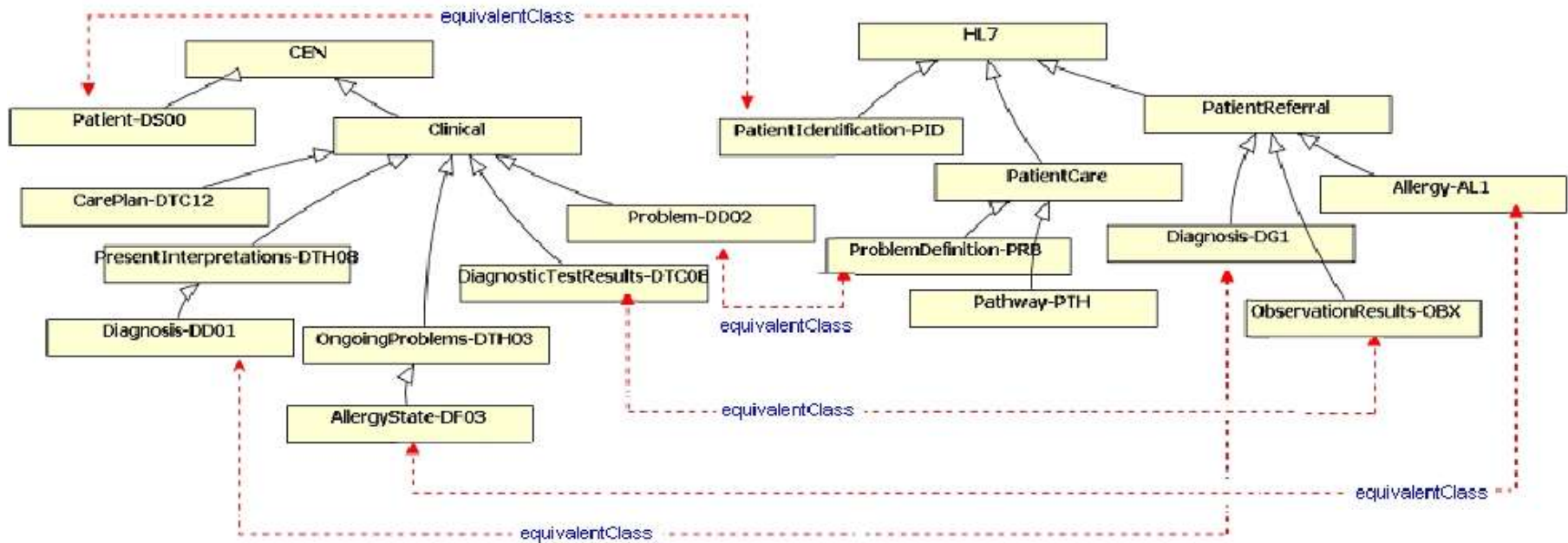
Ontology Mapping vs Schema Matching.

An example tool: MAFRA



# Ontology Mapping

Ontology Mapping is the process whereby two ontologies are semantically related at conceptual level, and the source ontology instances are transformed into the target ontology entities according to those **semantic relations**.





# Ontology Mapping

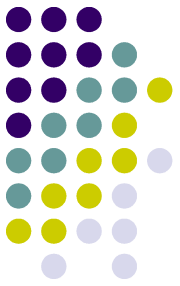
There are three dimensions of ontology mapping:

**Discovery:** manually, automatically or semi-automatically defining the relations between ontologies

**Representation:** A language to represent the relations between the ontologies

**Execution:** Changing instance of a source ontology to an instance of target ontology

# Discovery



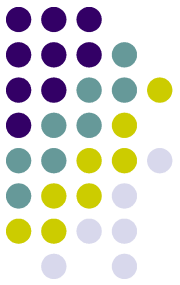
Mission: Find the related concepts/attributes of ontologies and the relation between them.

Need for Automatic Mapping:

Manually specifying schema matches is a time consuming, error-prone, and therefore an expensive process.

There is a rapidly increasing number of web data sources, and e-business to integrate which in turn shows the greatness of ontologies and data to be mapped.

A similar area in which lots of research done is  
“**Schema Matching**”



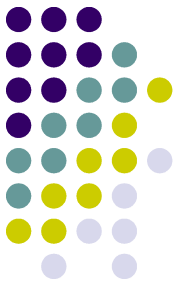
# Overview

Ontology Mapping

Schema Matching

Ontology Mapping vs Schema Matching.

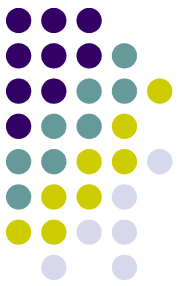
Example tool: MAFRA



# Schema Matching

Aims to provide a **matching** between database schemas

Match includes the mapping between the elements of two schemas that correspond to each other **semantically**



# Schema Matching

A particular representation:

Entity Relationship (ER) model, Object Oriented (OO) model, XML, Directed graphs

Mapping is a set of mapping elements each of which indicates that certain elements of schema S1 are mapped to certain elements in schema S2.

Each mapping may have a **mapping expression** which specifies the relation which maybe: simple relations over scalar (<, >, =), functions, ER style relationships, set oriented relationships For example:

Concatenate(Cust.FirstName, Cust.LastName) = Customer.Contact



# Schema Matching

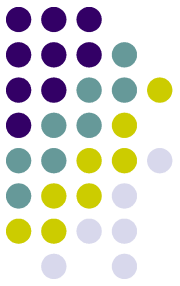
The result of mapping operation is “**match result**”

In general it is not possible to determine fully automatically matches since most schemas have semantics that effects the matching criteria but is not formally expressed or documented

Partial Structural map and Full Structural map

Matcher returns “**match candidates**”, which user accept, reject or change.

# Classification of schema matching approaches



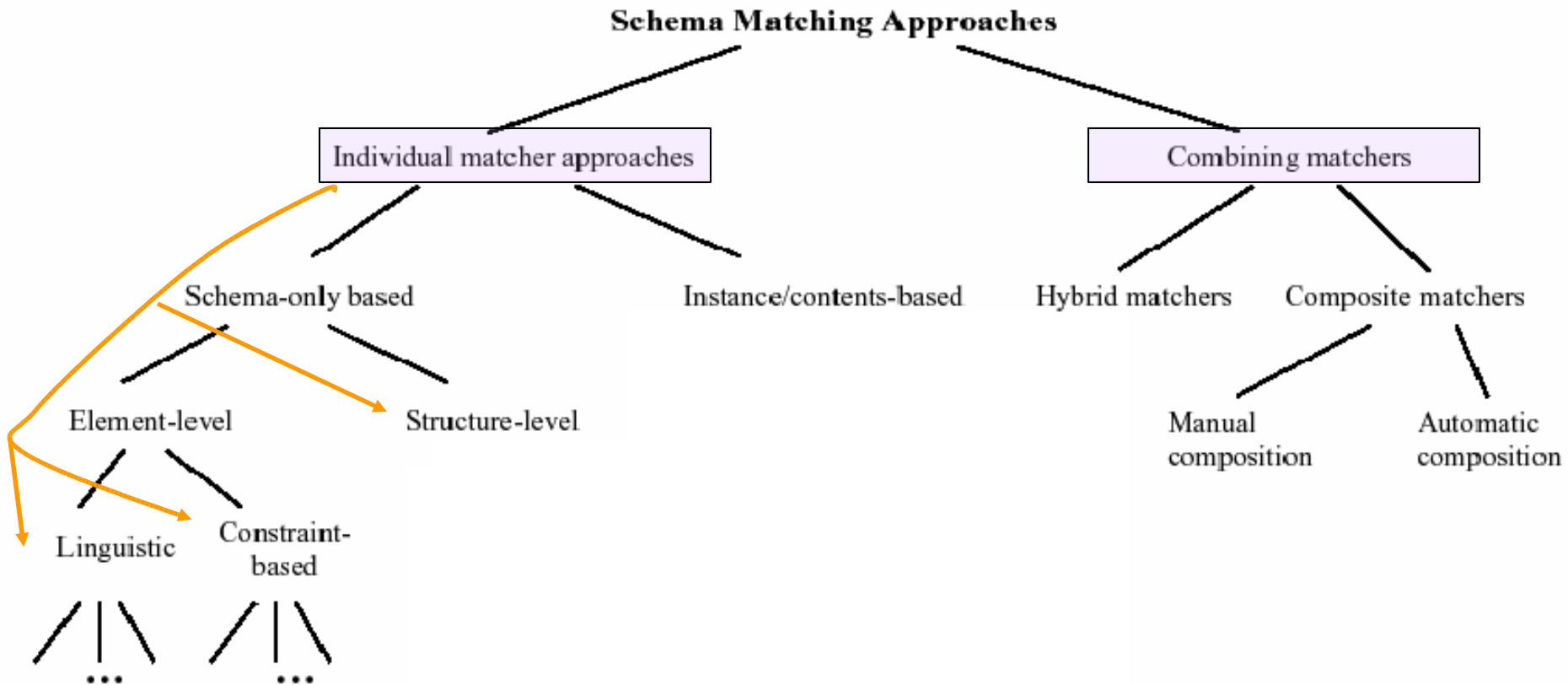
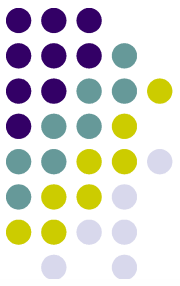
**Instance vs Schema:** consider the instances or only the schema

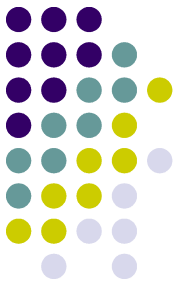
**Element vs Structure:** perform match for individual schema elements (such as attributes), or for combination of elements

**Language vs Constraint:** use textual names and descriptions, or the keys and relationships

**Matching Cardinality:** overall match result may relate one or more elements of one schema to one or more elements of other schema. (1:1, 1:n, m:n)

# Classification of schema matching approaches



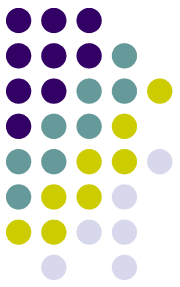


# Schema Level Matchers

Schema Level consider schema info, such as name, description, data type, relationship types (is-a, part-of), constraints, and schema structure

## **Element level**

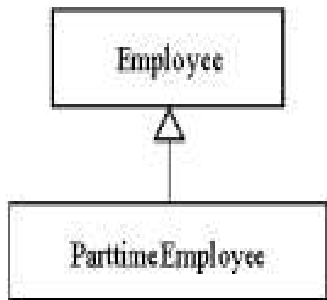
match considers only the atomic granularity elements of the schema such as attributes in xml schema or columns in relational schema



# Schema Level Matchers

## Structure level:

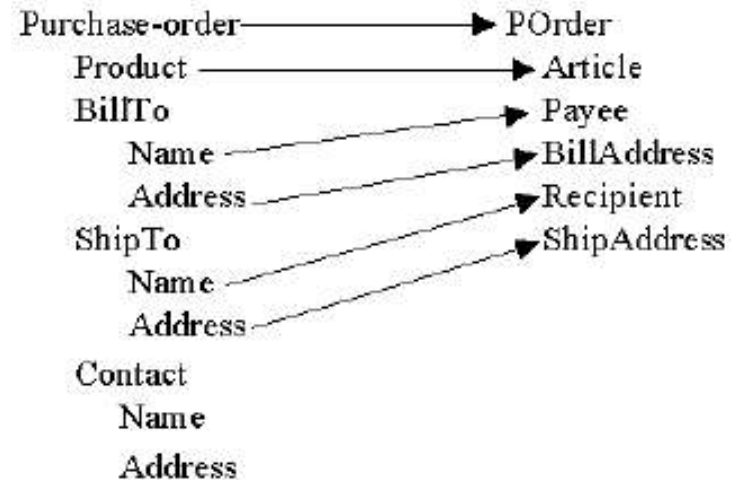
match refers to matching combinations of elements that appear together in a structure



III

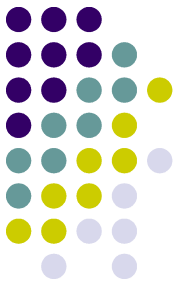


(a)



(b)

# Linguistic based approach



Linguistic based approaches use names and text to find semantically similar schema elements

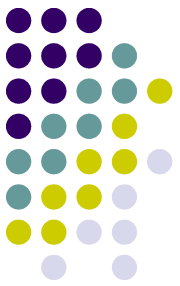
Equality of names

Equality of canonical name representations: CName = customerName

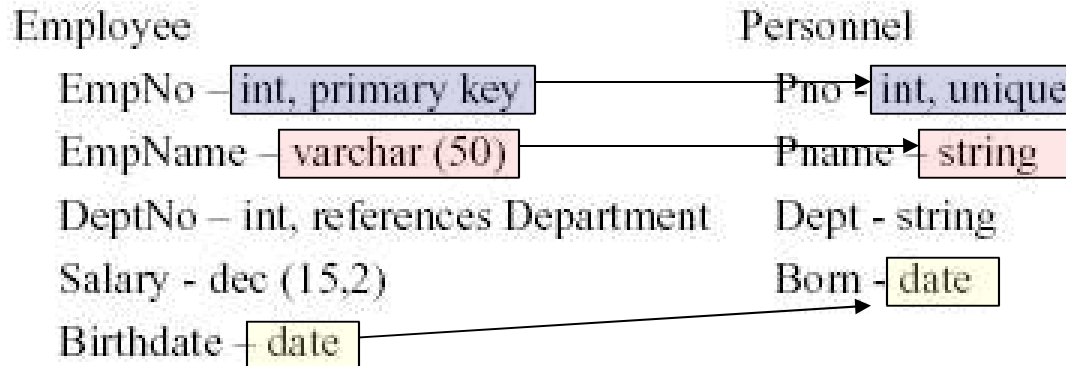
Equality of synonyms: car = automobile

Requires use of dictionaries (even multi-language), and taxonomies

Homonyms (words that are written in the same format but meaning different) are introduce problems



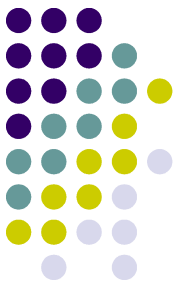
# Constraint based approach



Schemas has constraints to define data types, value ranges, uniqueness, optionality, relationship types and cardinalities.

Similarity is based on this constraints

Not so meaningful to use alone, but increases the reliability when used with other approaches.



# Matching Cardinality

An element in S1 can participate more than one match result between S1 and S2.

Or within an individual match result, one or more S1 elements can match to one or more S2 elements.

1:1, 1:n, n:1, (m:n)

	S1 element(s)	S2 element(s)	Matching expression
1:1 →	Price	Amount	Amount = Price
n:1 →	Price, Tax	Cost	Cost = Price*(1+Tax/100)
1:n →	Name	FirstName, LastName	FirstName, LastName = Extract (Name, ...)
m:n →	B.Title, B.PuNo, P.PuNo, P.Name	A.Book, A.Publisher	A.Book, A.Publisher = Select B.Title, P.Name From B, P Where B.PuNo=P.PuNo



# Overview

Ontology Mapping

Schema Matching

Ontology Mapping vs Schema Matching

An example tool: MAFRA

# Differences between Schema Matching and Ontology Mapping



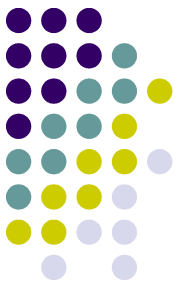
Database schema does not provide explicit semantics for their data, where ontologies does explicitly and formally

Database schemas are not sharable or reusable, usually they are defined over a specific database, whereas ontologies are by nature reusable and sharable

Ontology development is a more and more decentralized procedure

Database evolution should take into account the effects of each change on the data (addition of a new class), where in ontologies, the number of the knowledge representation primitives is much higher and more complex: cardinality constraints, inverse properties, transitive properties, disjoint classes, type-checking constraints

Ontology mapping is seems to be more reliable with the previous properties



# Mapping Representation

MAFRA (A.Maedche, N.Silva, B.Motik, R.Volz) and RDFT (C.Bussler, D.Fensel, B.Omalayenko) are two representation initiatives for mappings.

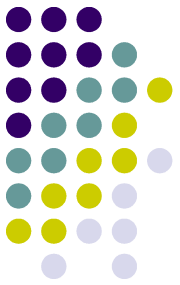
Both have similar logic to represent the mappings

Both uses “Bridges” to define the mapping between two schemas

Bridge establishes encapsulation of correspondences between entities from source and target ontology

Both defines a meta-ontology of bridges

Will be clear with MAFRA



# Overview

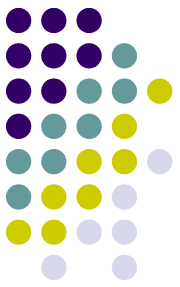
Ontology Mapping

Schema Matching

Ontology Mapping vs Schema Matching

Example tool: MAFRA

# MAFRA



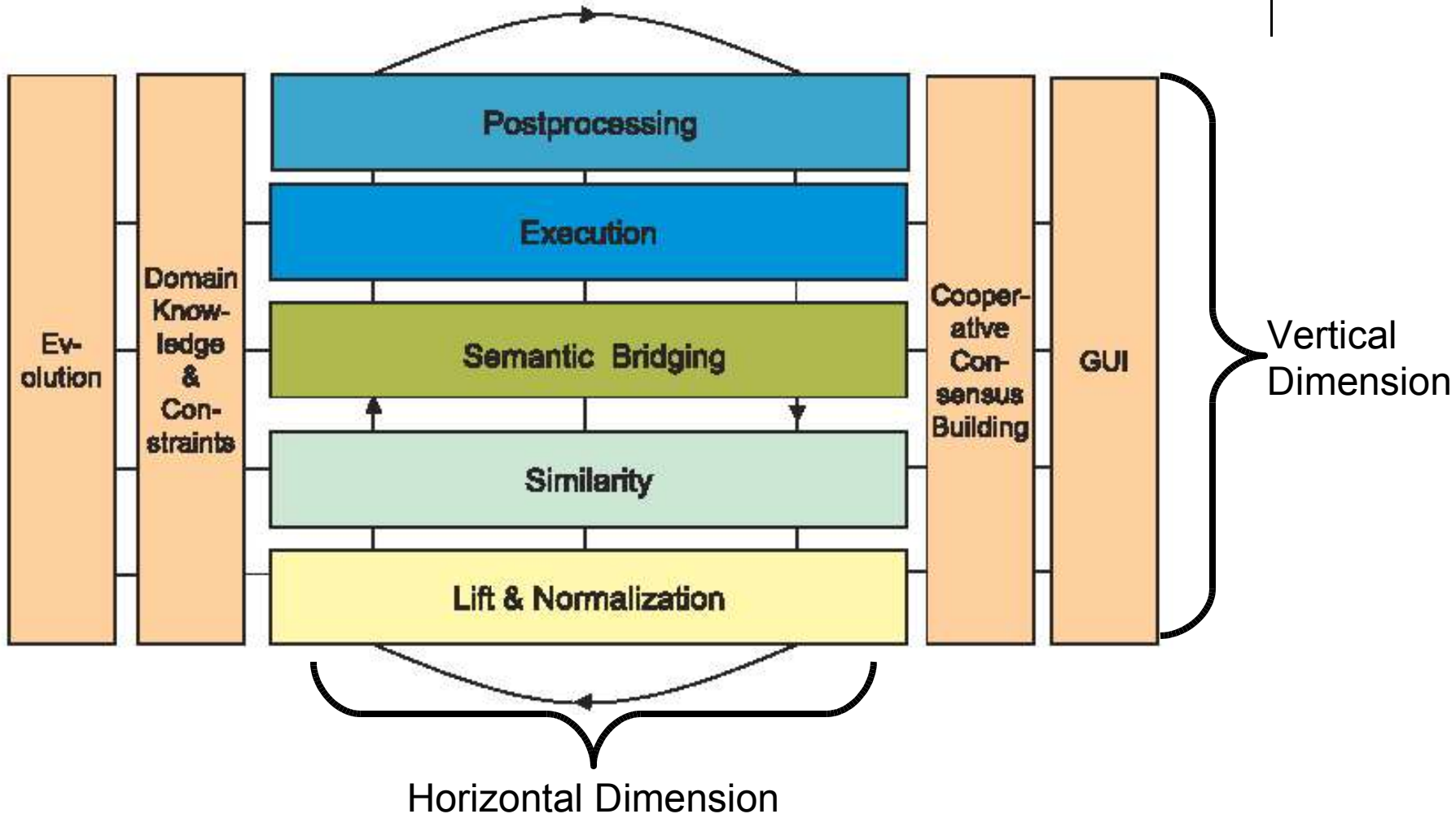
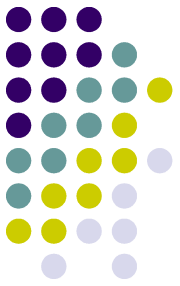
“A **MA**pping **FR**amework for Distributed Ontologies”,  
developed at Univ. Karlsruhe

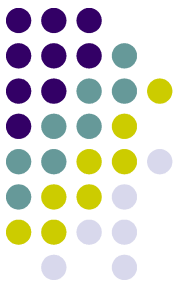
One of the main contributions is the definition of  
“**Semantic Bridges**” (SB) between ontologies which  
establishes correspondences between entities from  
source and target ontology.

Defines “**Semantic Bridge Ontology**” which is an  
ontology of mapping constructs.

Includes functionality for all of the three dimensions of  
ontology mapping (discovery, representation, execution)

# MAFRA Conceptual Architecture





# Horizontal Dimensions

**LIFT & Normalization:** Raise all data to be mapped to the same representation level, and normalize the strings (tokenization, expansion of acronyms...)

**Similarity:** Mapping discovery. Calculates the similarities according to several already proposed algorithms.

**Semantic Bridging:** Represent the mapping, will be explained in detail

**Execution:** Transform instances from source ontology to target ontology

**Post-Processing:** Takes the results of execution to check and improve the quality of transformation.

# Semantic Bridges



Has five dimensions:

**Entity dimension:** bridge may relate ontology entities such as concepts, relations, and attributes.

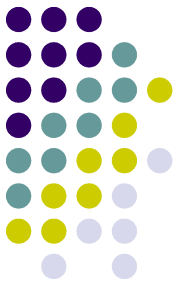
**Cardinality dimension:** Matching may be 1:1, 1:n and n:1.

**Structural dimension:** The way how elementary bridges may be combined into more complex bridge (specialization, alternatives, composition, abstraction)

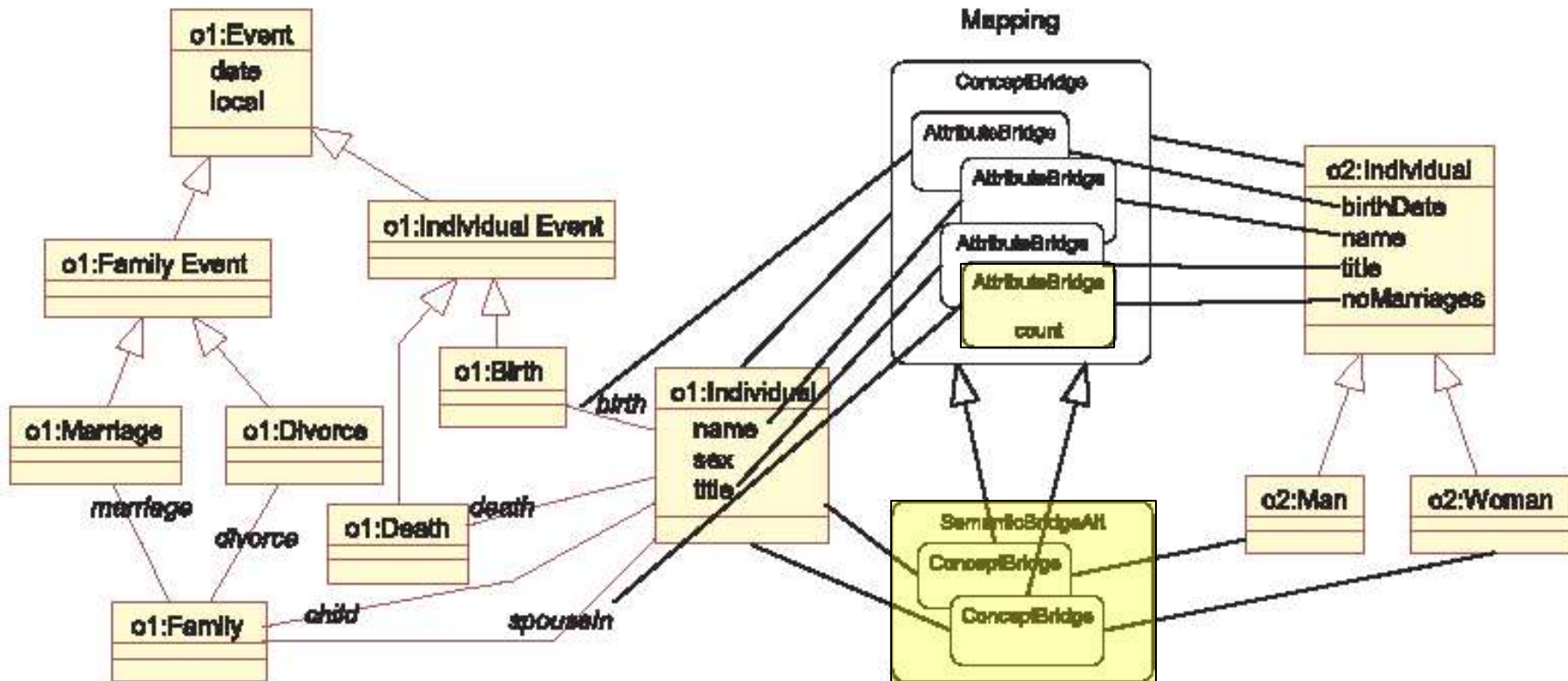
**Constraint dimension:** Controls the execution of bridge. Acts as conditions that must hold in order the transformation to take place.

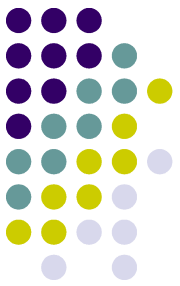
**Transformation dimension:** How the instances are transformed.





# MAFRA Mapping Example





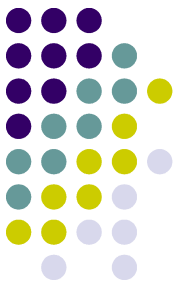
```
<Mapping rdf:ID="mapping">
  <relatesSourceOntology rdf:resource="&o1;" />
  <relatesTargetOntology rdf:resource="&o2;" />
  <hasBridge rdf:resource="#Individual-Individual" />
</Mapping>

<SemanticBridgeAlt rdf:ID="ManOrWoman">
  <hasBridge><Seq ordinal="1"><bridge rdf:resource="#Individual-Woman" /></Seq>
</hasBridge>
  <hasBridge><Seq ordinal="2"><bridge rdf:resource="#Individual-Man" /></Seq>
</hasBridge>
</SemanticBridgeAlt>

<ConceptBridge rdf:ID="Individual-Woman">
  <subBridgeOf rdf:resource="#Individual-Individual" />
  <relatesSourceEntity rdf:resource="#Individual" />
  <relatesTargetEntity rdf:resource="#Woman" />
  <whenVerifiedCondition rdf:resource="#isFemale" />
</ConceptBridge>

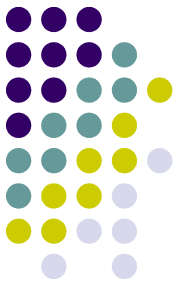
<ConceptBridge rdf:ID="Individual-Man">
  <subBridgeOf rdf:resource="#Individual-Individual" />
  <relatesSourceEntity rdf:resource="#Individual" />
  <relatesTargetEntity rdf:resource="#Man" />
</ConceptBridge>

<Condition rdf:ID="isFemale">
  <putServiceArgument>
    <MapArg><from>1</from><to rdf:resource="#sex" /></MapArg>
  </putServiceArgument>
  <putServiceArgument>
    <MapArg><from>pattern</from><to>F</to></MapArg>
  </putServiceArgument>
  <inService>CascadeAndMatch</inService>
</Condition>
```



```
<AttributeBridge rdf:ID="marriages">
  <relatesSourceEntity rdf:resource="#spouseIn"/>
  <relatesTargetEntity rdf:resource="#noMarriages"/>
  <accordingToTransformation rdf:resource="#countSpouses"/>
</AttributeBridge>

<Transformation rdf:ID="countSpouses"> <putServiceArgument>
  <MapArg><from>relation</from><to rdf:resource="#spouseIn"/></MapArg>
</putServiceArgument>
<mapTargetArgument>
  <MapArg><from>count</from><to rdf:resource="#noMarriages"/></MapArg>
</mapTargetArgument>
<inService>CountRelations</inService>
</Transformation>
```



# Thanx for Listening

**(Further) Questions?**

