

---

## ***Semantic Interoperability Centre Europe***

# **Study on Methodology**

Issue: Version 1.2

Date: 2009-06-02

Authors: Fraunhofer ISST / Jinit[



## Document Change History

Date	Version	Author	Change Details
2009-03-16	0.1	HA, AB, JE	Internal Draft
2009-03-23	0.2	HA, AB, JE	Initial Draft
2009-03-30	0.3	HA, AB, JE	Advanced Draft
2009-04-01	0.4	RFS	Comments
2009-04-09	0.9	HA, AB, JE	Pre-final Version
2009-04-14	0.95	KR, RFS, SM	Comments
2009-04-17	1.0	HA, AB, JE	Final Version
2009-04-19	1.1	HA, AB, JE	Reviewed Final Version
2009-04-29	1.15	Aldo Laudi	Comments
2009-06-02	1.2	HA, AB, JE	Revised Final Version

## Table of Contents

<b><i>Preface</i></b> .....	<b>6</b>
About SEMIC.EU .....	6
About IDABC .....	6
Conventions .....	6
<b><i>Management Summary</i></b> .....	<b>7</b>
<b><i>1. Introduction</i></b> .....	<b>8</b>
1.1 The Purpose of this Document .....	8
1.2 The Structure of this Document .....	9
<b><i>2. Semantic Conflicts</i></b> .....	<b>10</b>
2.1 Data-Level Conflicts .....	10
2.1.1 Data-Value Conflicts .....	10
2.1.2 Data-Representation Conflicts .....	10
2.1.3 Data-Unit Conflicts .....	10
2.1.4 Data-Precision Conflicts .....	11
2.1.5 Data-Language Conflicts .....	11
2.2 Schema-Level Conflicts .....	11
2.2.1 Naming and Conceptualisation Conflicts .....	11
2.2.2 Schema-Isomorphism Conflicts .....	11
2.2.3 Missing-Entity Conflict .....	12
2.2.4 Data Type Conflict .....	12
2.2.5 Generalisation Conflicts .....	12
2.2.6 Schematic Discrepancies .....	12
2.2.7 Hierarchy Conflicts .....	12
2.2.8 Foreign Key Conflicts .....	13
2.2.9 Constraints Conflicts .....	13
<b><i>3. Information Integration</i></b> .....	<b>14</b>
3.1 Schema Mapping .....	14
3.1.1 Value Correspondences .....	15
3.1.2 Mapping Situations .....	17
3.1.3 Mapping Example .....	19
3.1.4 Purposes .....	21
3.1.5 Mapping Tools .....	22
3.2 Schema Integration .....	22
3.2.1 Purposes .....	24

---

3.2.2	Schema integration tools.....	26
3.3	Schema Matching.....	26
3.3.1	Schema-based matching.....	26
3.3.2	Instance based matching .....	27
3.3.3	Purposes.....	28
3.3.4	Schema Matching Tools .....	28
3.4	Data Transformation .....	29
<b>4.</b>	<b><i>Controlled Vocabularies .....</i></b>	<b>30</b>
4.1	Types of Controlled Vocabulary.....	31
4.1.1	Glossaries and Code Lists.....	31
4.1.2	Taxonomies.....	34
4.1.3	Thesauri .....	37
4.1.4	Ontologies.....	39
4.1.5	Full-Fledged Ontologies .....	41
4.1.6	Summary of Purposes .....	44
4.2	Standard Languages for Controlled Vocabularies .....	45
4.2.1	Resource Description Format .....	46
4.2.2	OWL .....	47
4.2.3	Thesauri Standard and SKOS .....	49
4.2.4	Relationships to Controlled Vocabulary Types .....	50
4.3	Development of Controlled Vocabularies.....	51
<b>5.</b>	<b><i>Modularisation.....</i></b>	<b>55</b>
5.1	Modularisation Principles .....	55
5.1.1	Structured Data Types .....	55
5.1.2	Aggregation/Composition.....	56
5.1.3	Generalisation/Specialisation.....	57
5.1.4	Application Dependency.....	58
5.1.5	Domain Dependency.....	58
5.1.6	Domain Independency .....	59
5.1.7	Invariance .....	59
5.2	Reuse and Granularity.....	60
5.2.1	Bundling of Small Schemata .....	60
5.2.2	Parameterisation of Large Schemata.....	60
<b>6.</b>	<b><i>Conclusions.....</i></b>	<b>61</b>

## Table of Figures

Figure 1: Layers of interoperability .....	8
Figure 2: Schema mapping process.....	15
Figure 3: 1:1, N:1 and 1:N value correspondences .....	16
Figure 4: Higher-order correspondence .....	16
Figure 5: Higher-order correspondence .....	16
Figure 6: Mapping from structured to flat schema.....	17
Figure 7: Mapping from flat to structured schema.....	17
Figure 8: Mapping from a schema with a foreign key to schema without one .....	18
Figure 9: Mapping from a schema without a foreign key to one with a foreign key .....	18
Figure 10: Classification of mapping situations.....	19
Figure 11: Mapping from flat to structured schema.....	20
Figure 12: Logical relations .....	21
Figure 13: Integration of two schemata.....	24
Figure 14: Types of controlled vocabularies.....	30
Figure 15: Glossary fragment.....	32
Figure 16: Code lists .....	33
Figure 17: Tree of Porphyry.....	34
Figure 18: Taxonomic rank .....	36
Figure 19: Ontologies.....	40
Figure 20: Intensional and extensional knowledge .....	43
Figure 21: RDF and RDF Schema .....	46
Figure 22: OWL fragment in RDF .....	47
Figure 23: Activities of the Ontology Development Process.....	51
Figure 24: Conceptualisation tasks according to [FGJ97] .....	51
Figure 25: Questions for subclass relationship.....	53

## PREFACE

### About SEMIC.EU

SEMIC.EU (Semantic Interoperability Centre Europe) is an EU Project to support the data exchange for pan-European e-Government services. Its goal is to create a repository for interoperability assets that can be used by e-Government projects and their stakeholders. SEMIC.EU offers the following services for the public sector in Europe:

- 1 SEMIC.EU will provide access to interoperability assets that have been developed in previous governmental projects.
- 2 A clearing process will safeguard certain rules and standards to assure the quality of published assets.
- 3 Community features will be available on the platform, e.g. a forum to discuss best practices for the use of assets.
- 4 SEMIC.EU will invite stakeholders to seminars and workshops that are related to its activities.
- 5 SEMIC.EU offers coaching services for the creation and/or reuse of interoperability assets.

More information on SEMIC.EU can be found at: <http://www.semic.eu>.

SEMIC.EU is an action of IDABC. Contracted technical service providers for the project are: Jinit[ (main contractor), Fraunhofer ISST, GEFEG, and France Telecom R&D.

### About IDABC

IDABC stands for Interoperable Delivery of European e-Government Services to public Administrations, Business, and Citizens. It takes advantage of the opportunities offered by information and communication technologies to encourage and support the delivery of cross-border public-sector services to citizens and enterprises in Europe and to improve efficiency and collaboration among European public administrations.

The programme also provides financing to projects addressing European policy requirements, thus improving cooperation among administrations across Europe. National public-sector policy makers are represented in the IDABC programme's management committee and in many expert groups. This makes of the programme a unique forum for the coordination of national e-Government policies.

<http://ec.europa.eu/idabc>

### Conventions

The type styles shown below are used in this document to emphasize parts of the text.

Times New Roman – 11 pt.: Standard body text

*Times New Roman – 11 pt. Italic:* Citations

The requirements level indicators are fully aligned to “RFC2119 - Key words for use in RFCs to Indicate Requirement Levels” and are used as follows:

**MUST** means that this policy element or requirement is to be fulfilled without exception.

**SHOULD** indicates an optional policy element / requirement that may be fulfilled if desired.

## MANAGEMENT SUMMARY

The objective of SEMIC.EU is the publication and reuse of Semantic Interoperability Assets for pan-European data exchange among public administrations. This data exchange has to preserve the semantics, meaning that the sender and receiver of a message shall have an identical understanding of the data exchanged.

If the sender and receiver use different data structures when exchanging data, semantic conflicts occur. The best way to avoid semantic conflicts is the mandatory use of standardised data schemata by all participants of a data exchange. Due to the autonomy of the Member States, however, it is not possible to enforce the use of common schemata on a pan-European level. Therefore, semantic conflicts will occur and have to be handled appropriately.

This study presents state-of-the-art methodologies for treating the different types of semantic conflicts, i.e. detecting and resolving these conflicts or even avoiding them by reusing existing solutions. These methodologies comprise schema mapping, schema integration, schema matching, controlled vocabularies and modularisation principles.

Schema mapping techniques are introduced that resolve different types of semantic conflicts by detecting correspondences between semantically equivalent sub-structures of two different schemata, identifying mappings between these schemata and deriving data transformations for data conforming these schemata.

Schema integration techniques are presented that are applied, when – as usual in a pan-European context – a large number of different schemata is involved. In this case, the data exchange is performed via an intermediary, i.e. an integrated schema is created by applying these schema integration techniques.

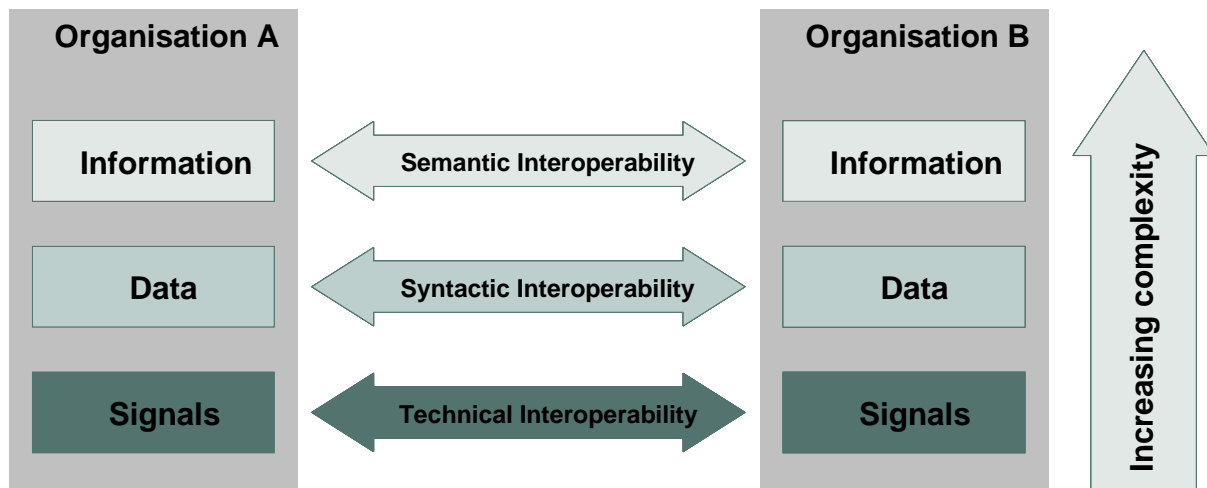
Neither the development of schema mappings nor the integration of schemata, particularly for large and complex schemata, can be performed without applying methods of schema matching. The study describes techniques for the (semi-)automatic detection of semantic equivalence relations between schema elements.

Controlled vocabularies are one of the main methods to facilitate schema matching and the handling of semantic conflicts. Because the aim of employing controlled vocabularies is to clarify the semantics of terms used in schemata, they constitute an appropriate means to detect and resolve semantic conflicts. Different types of controlled vocabularies are presented in this study, and it is demonstrated that the greater the complexity and expressiveness of a controlled vocabulary, the higher the number of types of semantic conflicts that can be covered. In addition, concrete vocabulary languages are presented and related to the type of controlled vocabulary they implement.

Finally, the study introduces modularisation principles for an appropriate structuring of schemata. Applying these principles supports the detection of semantic conflicts, and minimises semantic conflicts by enabling the reuse of existing modular assets and existing solutions for resolving semantic conflicts.

## 1. INTRODUCTION

The fundamental objective of SEMIC.EU is semantic interoperability. The important layers of interoperability in the context of SEMIC.EU are shown in Figure 1.



*Figure 1: Layers of interoperability*

Technical interoperability, i.e. the interoperability on the protocol level, and organisational interoperability (not depicted in the above figure), i.e. the operability between communication partners related to underlying business processes and related agreements, are out of the scope of SEMIC.EU. Syntactic components are only subsumed under the heading of semantic interoperability (see [Hel06]) if they are indispensable supplements to semantic elements such as terminologies, thesauri, and mapping rules, and hence support semantic interoperability.

The fundamental concept that guarantees semantic interoperability in SEMIC.EU is what is called Semantic Interoperability Assets (as described in [FI08]). These assets comprise different components, in particular schemata defining the structure of the data exchanged. If different schemata are used by the respective communication partners, semantic conflicts, i.e. a different understanding of the meaning of the data, occur. These conflicts have to be identified and dealt with.

The most adequate way to avoid semantic conflicts effectively is to develop a standard (e.g. a common schema for eHealth data) that is accepted and used by all parties involved in mutual data exchange. In a pan-European context, however, the high degree of autonomy of the Member States precludes prescription of the use of such schemata. For example, national standards that incorporate the legal requirements to be obeyed in one Member State may lead to incompatibility with standards used by another.

As a consequence, semantic conflicts will definitely occur on the pan-European level. This problem also arises regularly in federated political structures, i.e. on a national or even regional level. These conflicts need to be detected and – as far as possible – be resolved.

### 1.1 The Purpose of this Document

The study presents methodologies for achieving the essential goals of SEMIC.EU, i.e. fostering reuse of Semantic Interoperability Assets and safe-guarding semantic interoperability by an appropriate handling of semantic conflicts. The preferred approach is to avoid semantic conflicts by a proper



design of Semantic Interoperability Assets and the reuse of these assets and their artefacts, respectively. Moreover, it is shown, how any remaining semantic conflicts can be identified and resolved, whenever possible. This is done by introducing and describing detailed methods and techniques facilitating establishing a relationship between the types of semantic conflicts and the appropriateness of employing a specific method for dealing with it.

The study presents research results from a theoretical-conceptual point of view, i.e. abstracting from concrete projects and applications, and connects them to the constraints and related requirements to ensure semantic interoperability when exchanging data. Additionally, this study focuses on the current state of research and the discussion of the methods and techniques used to ensure semantic interoperability with a clear relation to purposes. However, a full consideration of legal, political, and organisational factors is beyond its scope.

## **1.2 The Structure of this Document**

In the second section, the document introduces a classification and description of different types of semantic conflicts, which then will be related to methodologies and techniques for avoiding, detecting, or resolving them. In the third section, information integration - the core discipline for solving semantic conflicts between data based on different schemata - is presented. Thereby, schema mapping, schema matching, and schema integration are described in detail. The fourth section deals with controlled vocabularies, which support the information integration process. Several types of controlled vocabularies are introduced and compared according to their respective suitability for handling semantic conflicts. The fifth section discusses concepts and principles of modularisation applied to schemata within Semantic Interoperability Assets with regard to avoiding semantic conflicts and reuse of Semantic Interoperability Assets. The findings of the study are summarised in the conclusions.

## 2. SEMANTIC CONFLICTS

There are several descriptions and different classifications of semantic conflicts in the corresponding literature (see [SK92], [W03], [PR04], [LN06], [LN07]). As in [PR04], the classification and description presented here distinguishes between data-level conflicts and schema-level conflicts. These conflicts are related to methodologies and techniques for avoiding, resolving, and detecting these conflicts, and references are provided to the sections in which these methodologies and techniques are described.

### 2.1 Data-Level Conflicts

*“Data-level conflicts are differences in data domains caused by the multiple representations and interpretations of similar data.”* [PR04]

#### 2.1.1 Data-Value Conflicts

Data-value conflicts occur due to different meanings for the same designation, known as homonyms, *“e.g., in soil suitability analysis databases, the data value “suitable” in one database may mean that the soil in a particular area is suitable for road construction, while the same value in another database may indicate that the soil is suitable for sewage disposal”* [PR04].

Data-value conflicts also occur concerning synonyms, e.g. in Germany “England” is often used as a synonym for “Great Britain”, which is not the case in Great Britain

Data-value conflicts can be avoided, resolved or detected by means of controlled vocabularies (see section 3). Generally, homonyms may indicate other conflicts, e.g. naming conflicts. The first example above shows that the “suitability” in one database differs from a quality indicated by the same word the other, i.e. they denominate different concepts, which correspond to a naming and conceptualisation conflict (see section 2.2.1).

#### 2.1.2 Data-Representation Conflicts

The same interpretation of data may be impeded by a different representation. A common representation conflict is the use of different representations for dates, *“e.g. a date can be represented as 06-30-2005 in one database, as 30-06-2005 in another and as 30-Jun-2005 in a third”* [TTPL07].

Different syntactical representations, e.g. different separators in lists or different character sets, can also be interpreted as representation conflicts.

A third kind of representation conflicts occurs when different fonts are used, e.g. Latin and Greek.

Detecting data representation conflicts can be supported by schema matching (see section 3.3). Representation conflicts can be resolved by converting the output data using an appropriate conversion function.

#### 2.1.3 Data-Unit Conflicts

Data-unit conflicts occur concerning different unit systems, e.g. the sender provides heights in centimetres, whereas the receiver expects heights being provided in inches.

Data unit conflicts can be resolved by applying conversion functions within the schema mapping process (see section 3.1)

#### 2.1.4 Data-Precision Conflicts

Data-precision conflicts occur when different scales are used, e.g. “high”, “medium”, “low” versus “very high”, “high”, “medium”, “low” and “very low”, or when the sender provides numerical values with a different number of decimal places than needed by the receiver.

Data-precision conflicts can be resolved when there is a reasonable mapping from the scale of the sender to the scale of the receiver and a related data transformation. Generally, this is only possible when the precision needed by the receiver is lower than the precision provided by the sender.

#### 2.1.5 Data-Language Conflicts

In the case of multilingualism, denominators are different for almost every concept when expressed in different languages. Moreover, for concepts that are more abstract, a translation for a denominator is not always available that exactly preserves the semantics.

An in-depth treatment of multilingualism in the SEMIC.EU context is provided in [FI08].

Data-value conflicts can be resolved or detected by means of controlled vocabularies, e.g. appropriate controlled vocabularies support the translation of data values (see section 3).

### 2.2 Schema-Level Conflicts

Schema-level conflicts comprise conflicts caused by varying names for equivalent information entities, different conceptualisations of the domain (see [FI08]), and by differences in structuring schemata.

#### 2.2.1 Naming and Conceptualisation Conflicts

Naming conflicts occur in regard to the relation between symbols/names and concepts:

- **Synonyms.** Two different symbols or names are synonyms if they denominate the same concept, e.g. the tag “publication” in one schema and the tag “paper” in another both designate “research article”
- **Homonyms.** Homonyms are symbols or names that may activate different concepts, usually depending on the context in which they are used, e.g. the element or attribute “id” designates different identifiers in different schemata.
- **Hypernyms/Subsumption.** A symbol or term is a hypernym of another if it is a broader term for it, e.g. “person” is a hypernym of “man” as well as of “woman”.
- **Overlapping.** If two different symbols or names refer to different sets of things, but the intersection of these sets isn’t empty, an overlapping conflict is present.

Detecting naming conflicts is supported by schema matching (see section 3.3). The resolution of naming conflicts is achieved by applying schema mapping (see section 3.1), but resolving naming conflicts is not possible in all cases, particularly subsumption and overlapping.

#### 2.2.2 Schema-Isomorphism Conflicts

Schema-isomorphism conflicts occur when sender and receiver describe the same concept by different sets of attributes. In terms of set operations, these sets of entities are not setoperation-compatible [SK92].

Resolution of these conflicts may be achieved by applying schema mapping (see section 3.1), e.g. by appropriate transformations, but is not possible in all cases.

### 2.2.3 Missing-Entity Conflict

Missing-entity conflicts occur when entities in the source or in the target schema occur without equivalent entities in their opposite counterpart, e.g. an attribute in the source schema lacks a corresponding attribute in the target schema or an attribute in the target schema has not a corresponding attribute in the source schema.

Detecting these conflicts is supported by schema matching techniques (see section 3.3). Resolving of these conflicts may be achieved by applying schema mapping (see section 3.1), but is not possible in all cases, e.g. for a mandatory attribute in a target schema without an equivalent attribute in the source schema there is no adequate mapping.

### 2.2.4 Data Type Conflict

Data type conflicts occur, when semantically equivalent entities have different data types, e.g. representation of numerical values as integers or as double/real.

Detecting these conflicts is supported by schema matching (see section 3.3). Resolving of these conflicts may be achieved by applying schema mapping techniques (see section 3.1), but depends on the applicability of appropriate conversion functions. For example, an integer can be transformed to a real number, but not vice versa.

### 2.2.5 Generalisation Conflicts

Generalisation conflicts occur when different choices of generalisation of concepts are used, e.g. when one schema contains the concept of a person, whereas the other schema distinguishes between men and women and therefore has different concepts for them.

Detecting these conflicts is supported by schema matching (see section 3.3) utilising controlled vocabularies (see section 3). Resolving of these conflicts may be achieved by applying schema mapping (see section 3.1), but is not always possible.

### 2.2.6 Schematic Discrepancies

Schematic discrepancies occur when the two schemata have different logical structures, e.g. an item is modelled as an element in one XML schema, whereas it is modelled as an attribute in the other XML schema.

Resolving of these conflicts is achieved by applying schema mapping techniques (see section 3.1).

### 2.2.7 Hierarchy Conflicts

These conflicts occur when the two schemata have different levels of hierarchy, e.g. a source XML schema is flat whereas the target schema is structured or vice versa.

Resolving of these conflicts is achieved by applying schema mapping techniques (see section 3.1).

### 2.2.8 Foreign Key Conflicts

These conflicts occur when a source XML schema uses foreign keys whereas the target schema does not use foreign keys or vice versa.

Resolving of these conflicts is achieved by applying schema mapping techniques (see section 3.1).

### 2.2.9 Constraints Conflicts

These conflicts occur when source XML schema and target schema differ concerning constraints, e.g. in the cardinality imposed on an attribute.

Detecting these conflicts is supported by schema matching (see section 3.3). Resolving of such a conflict may be impossible, e.g. when the source accepts a higher cardinality than the target schema.

The classification and description of the semantic conflicts presented above is the foundation for detecting and resolving these conflicts. Whereas some of the conflicts, e.g. the representation conflict caused by different date representations, are quite easy to detect and resolve, others are much more difficult to detect and to resolve, in particular, when more than two different schemata are involved.

The following sections introduce different approaches and methodologies that, depending on the specific conditions in the application scenario, provide solutions for detecting and resolving, or even avoiding semantic conflicts. Usually, depending on the complexity of the application scenario and the variety of conflicts implied, several or even all of these methodologies will be applicable within the same scenario, e.g. modularisation and reuse for avoiding semantic conflicts, and schema matching – supported by controlled vocabularies –, schema integration and schema mapping for detecting and resolving the remaining conflicts. The presentation of the methodologies starts with a deeper inspection of schema mapping, schema integration, and schema matching.

### 3. INFORMATION INTEGRATION

Methods of information integration facilitate semantic interoperability among distributed and heterogeneous (information) systems, in particular, information integration provides methods and techniques to resolve the semantic conflicts implied by the heterogeneity of these systems.

Methods of information integration include

- schema mapping, i.e. the mapping of one schema onto another and deriving transformation instructions for data conforming these schemata
- schema integration, i.e. the integration of several heterogeneous schemata in one integrated schema
- schema matching for (semi-)automatically detecting correspondences between equivalent elements within different schemata to support schema mapping as well as schema integration
- data integration

In order to determine the purposes of these methods with respect to the handling of semantic conflicts in a unified way, this study will use the tabular form shown in Table 1. The first part of the table states the general purpose of the method in a condensed form. For those who want to have an overview with respect to the conflict types introduced in section 2, the second part indicates (together with a short description) whether the handling of a conflict type is supported. This tabular form will also be used in section 4 for determining general and conflict-related support by controlled vocabularies.

general	
<i>general characteristic and purpose of methodology</i>	
conflict-related	
conflict type	support
<i>one row for each conflict type</i>	<i>short description of conflict handling in case of support</i>

*Table 1: Tabular form for purpose tables*

#### 3.1 Schema Mapping

Schema mapping is used when data exchange is based on different, heterogeneous schemata [LS08].

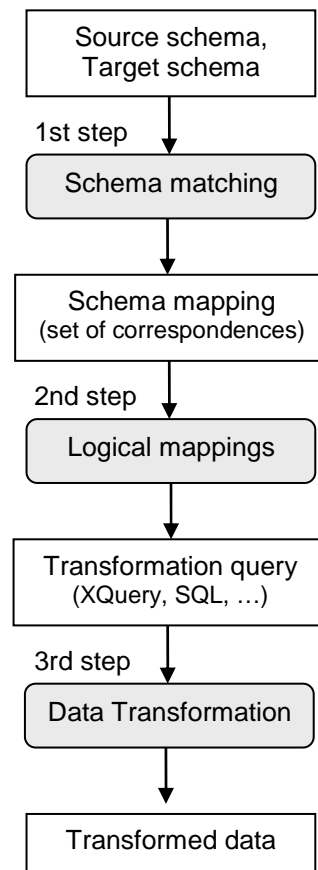
The general purpose of schema mapping is to resolve schema-level semantic conflicts to ensure that a data exchange based on different source and target schemata preserves its semantic.

On the one hand, schema mapping designates the correspondences between semantically equivalent entities of different schemata. On the other hand, (see Figure 2) schema mapping designates the process of

1. identifying correspondences between elements of two different schemata, a source schema and a target schema, by (at least for complex schemata) applying schema matching techniques (see section 3.3)
2. deriving consistent logical mappings (an example at the end of this subsection will illustrate this) considering different mapping situations

3. deriving corresponding instructions for transforming data that conforms to the source schema into data that conforms to the target schema

In the case of two XML schemata, schema mapping results in an XSLT or XQuery program that transforms XML documents that conform to the source schema into semantically equivalent XML documents that conform to the target schema.



*Figure 2: Schema mapping process*

### 3.1.1 Value Correspondences

Schema mapping is based on value correspondences that determine how data values of source attributes are mapped to data values of target attributes. Value correspondences exist between attributes that designate the same concept and capture the same information.

The simplest case is a 1:1-correspondence without any transformation of the data value. Transformations may be necessary, e.g. to cope with different date formats or in case of different units (see section 2.1). For two XML schemata, attributes of type string or integer are connected by value correspondences (see Figure 3).

1:N correspondences connect one source attribute with more than one target attribute, N:1 correspondences connect more than one source attribute with one target attribute. Both 1:N and N:1 correspondences are schema-isomorphism conflicts. In both cases, these conflicts can be resolved by functions that define appropriate data mappings (see Figure 3). The 1:1 correspondence in Figure 3 would not be an allowed correspondence when “name” on the target side is expected to contain “first name” and “last name”. This case would correspond to an irresolvable subsumption conflict, whereas

the N:1 as well as the N:1 correspondence shown in Figure 3 involve resolvable subsumption conflicts.

M:N-correspondences – though theoretically possible – have minor practical relevance.

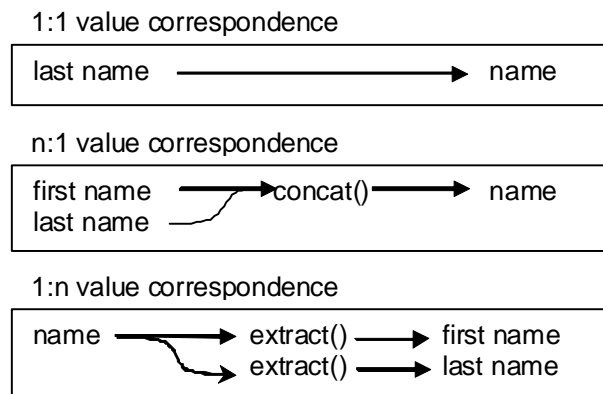


Figure 3: 1:1, N:1 and 1:N value correspondences

Value correspondences cannot cope with generalisation conflicts. Higher-order correspondences that connect entities of different types, e.g. an element and an attribute in an XML schema, address this problem.

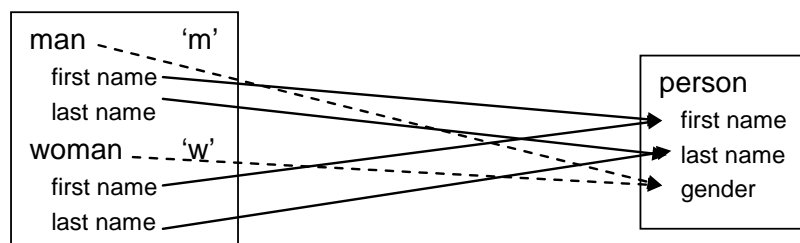


Figure 4: Higher-order correspondence

To resolve the related generalisation conflict, a data transformation based on the correspondence illustrated in Figure 4 has to provide values for the gender attribute depending on the source element to which the data is attached.

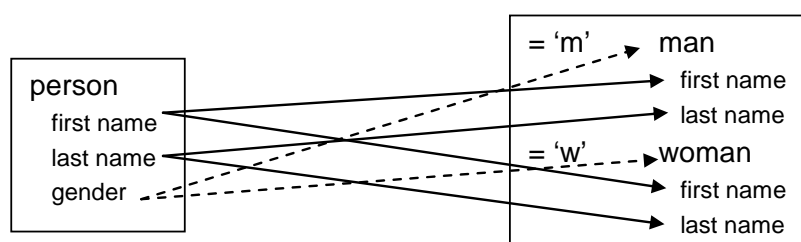


Figure 5: Higher-order correspondence



To resolve the related generalisation conflict, a data transformation based on the correspondence illustrated in Figure 5 has to attach a person to different target elements depending on the value of the gender attribute in the source data.

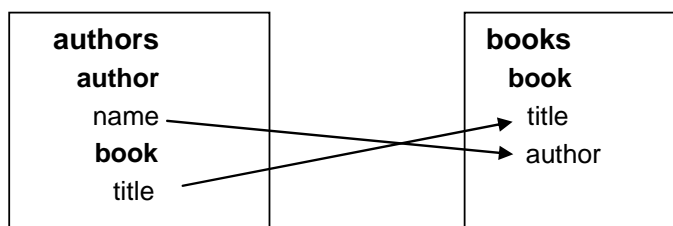
As these examples show, mappings usually cannot be reversed.

### 3.1.2 Mapping Situations

There are various mapping situations involving different semantic conflicts. The four “standard” situations described below involve hierarchy and foreign-key conflicts.

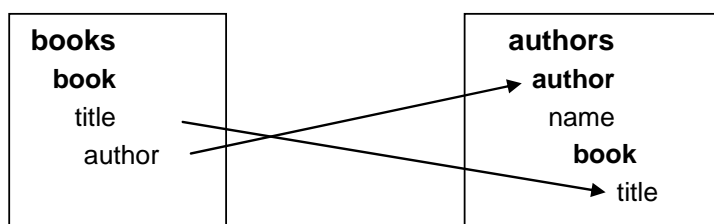
Two typical mapping situations are the mapping from a structured to a flat XML schema and vice versa, both involving a hierarchy conflict.

Figure 6 shows a mapping from a structured to a flat XML schema. Whereas an author only occurs once in an XML document that conforms to the source schema, in the transformed XML document the number of occurrences depends on the number of his books, what, in order to resolve the hierarchy conflict, has to be considered in a related transformation.



*Figure 6: Mapping from structured to flat schema*

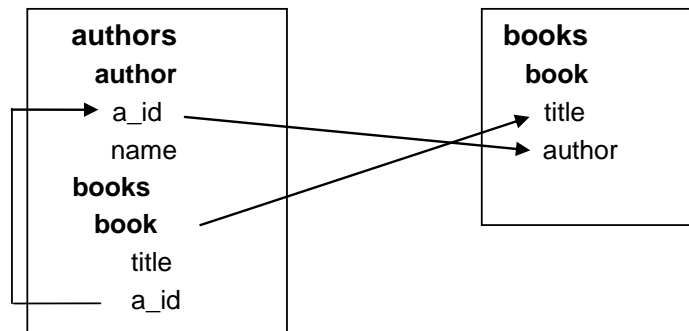
For the mapping from a flat to a structured XML schema shown in Figure 7, to resolve the hierarchy conflict, the related transformation has to ensure that each author only occurs once in the target document and that the assignment of authors to books from the source document is preserved.



*Figure 7: Mapping from flat to structured schema*

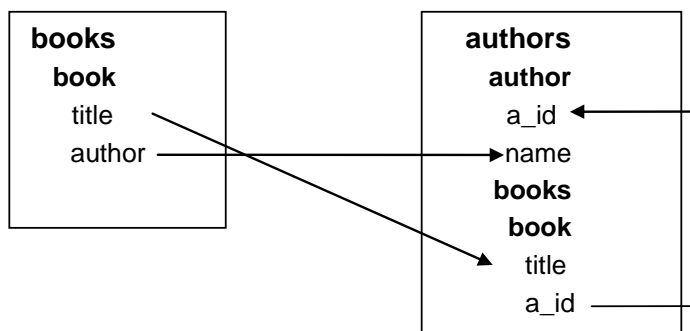
In both cases, an Xquery transformation with nested loops can be defined or generated by a schema mapping tool that generates the correct relations between authors and books.

Two other typical mapping situations concern the existence of primary keys.



*Figure 8: Mapping from a schema with a foreign key to schema without one*

Figure 8 shows a mapping from the source schema containing a foreign-key relation to a flat target schema, thus involving a foreign-key conflict [Cab09]. The corresponding data transformation has to ensure that the authors are related to the books according to the foreign-key relation in the source data.



*Figure 9: Mapping from a schema without a foreign key to one with a foreign key*

Figure 9 shows a mapping from a flat source schema to a target schema containing a foreign-key relation, which represents another foreign key conflict. The corresponding data transformation has to ensure that each author only occurs once in the target document. Moreover, keys have to be generated that relate authors and books preserving the assignment of authors to books from the source document. This is achieved by using what are known as Skolem-functions, which ensure the generation of unique identifiers [PVM+02].

Further mapping situations are missing correspondences, either on the source or on the target side, corresponding to a missing entity conflict. If an attribute of the target schema without any correspondence is a key or a foreign key, has to be unique, or is not optional, values for this attribute have to be generated, or default values have to be defined to resolve the conflict. Otherwise, the transformation yields target data that does not conform to the target schema. Whether the assignment of reasonable values or reasonable default values is possible depends on the specific context. In particular, mandatory values usually cannot be provided as default values and usually they cannot be derived from other values. This implies that not in each mapping situation an appropriate mapping is possible.

[LN06] provides a comprehensive classification of mapping situations corresponding to different semantic conflicts, shown in Figure 10.

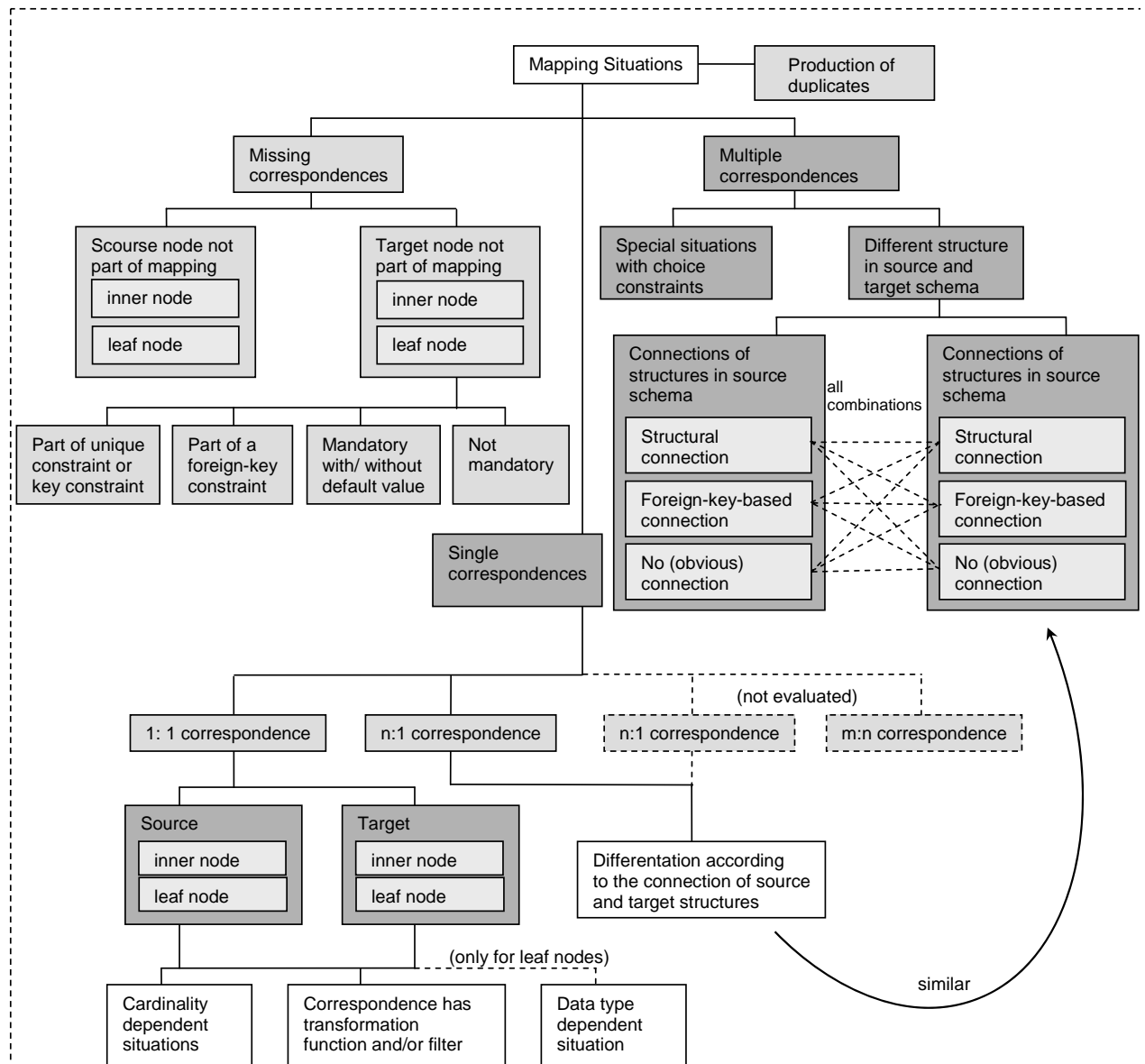
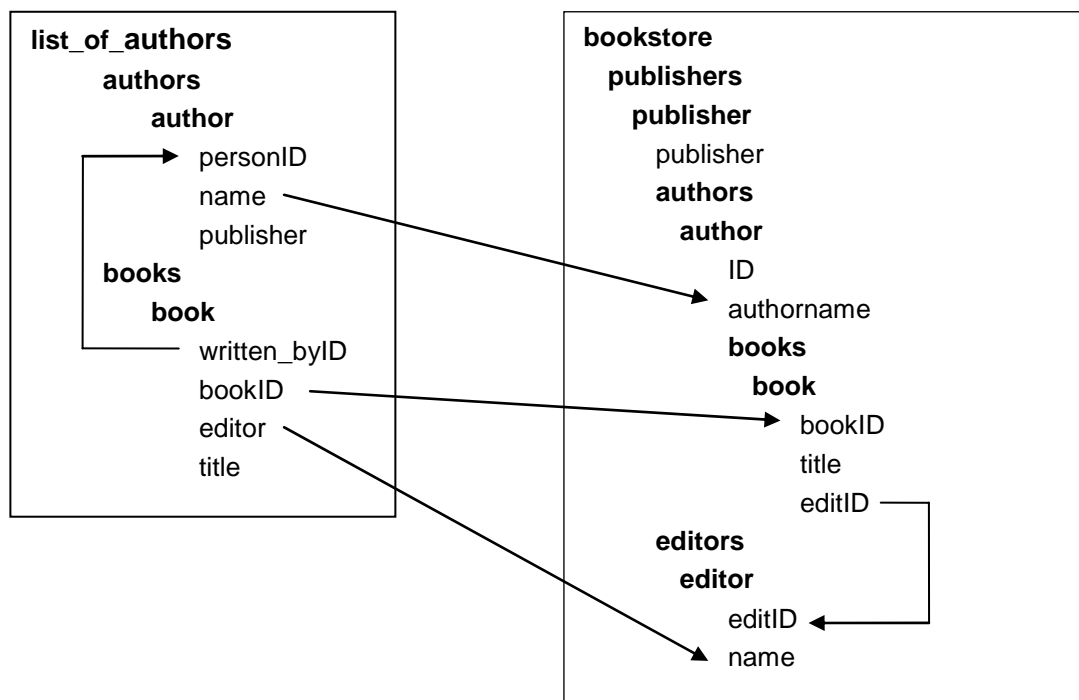


Figure 10: Classification of mapping situations

### 3.1.3 Mapping Example

How the logical mappings are derived is illustrated by applying the algorithm implemented in the Clio research prototype [PVM+02] to an example adapted from an example provided in [LN07]. This example (see Figure 11) contains a mapping from a flat XML schema to a structured XML schema, both of which contain foreign keys. Only three value correspondences are considered for deriving the logical mappings.



*Figure 11: Mapping from flat to structured schema*

In the first step, paths within each of the schemata involved are determined. In a flat schema, each of the elements below the root corresponds to a path. In a structured schema, each chain of elements related by a parent / child-relation is a path.

Resulting paths are P1: authors and P2: books in the source schema, and P3: publishers, P4: publishers – authors, P5: publishers – authors – books and P6: publishers – editors in the target schema.

Then for each path where the last element is in a foreign-key-relation to another element, this element is added to the chain as new last element.

As a result, P2 is extended to P2: books – authors and P5 is extended to P5: publishers – authors – books – editors.

Attributes of the schema are considered to be in a semantic association, called a logical relation (LR), when they lie within the same path. Hence, for each path there is a corresponding logical relation.

Now source logical relations are mapped to target relations. To do this, combinations composed of one source logical relation and one target logical relation each are built. In the next step, the combinations where the source of a correspondence is within the source logical relation, but the target of the correspondence is not in the target logical relation, and vice versa, are ruled out.

Only LR1 → LR4 and LR2 → LR5 remain. These logical relations determine two different logical mappings, the first one a transformation of authors with or without books, and the second one a transformation of authors with books. A domain expert has to decide which of the logical mappings will be used as a basis for the transformation.

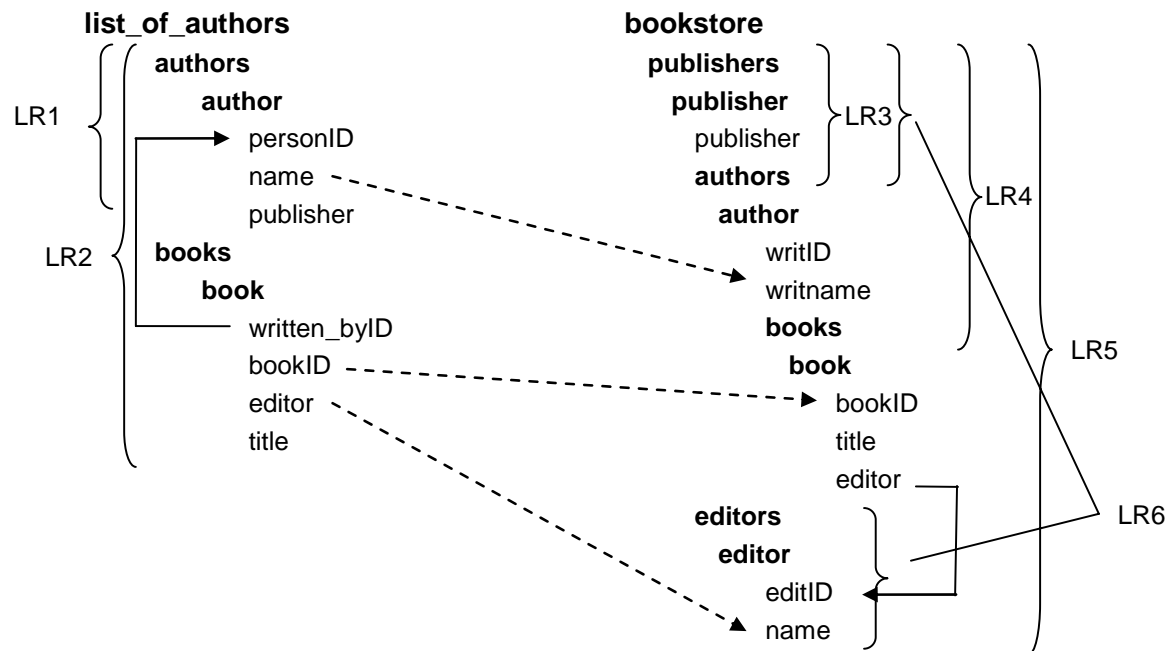


Figure 12: Logical relations

In the third step, based on the logical mapping chosen the data transformation instructions are generated, in case of XML schemata an XSLT or an XQuery program.

### 3.1.4 Purposes

general			
The general purpose of schema mapping is to resolve schema-level semantic conflicts to ensure that a data exchange based on different source and target schemata preserves its semantic.			
conflict-related			
conflict type		support	
data level	synonyms	+	transformation based on glossaries
	homonyms	-	not resolvable
	data representation	+	conversion function
	data unit	+	conversion function
	data precision	o	conversion function, conversion from lower to higher precision is not possible
	data language	+	translation based on controlled vocabularies
schema level	synonyms	+	building value correspondences
	homonyms	+	no correspondence

	hypernyms	o	appropriate functions, e.g. extract and concat, not always possible (see generalisation)
	overlapping	o	generally no correspondence (see 2.2.1), as no semantic equivalence
	schema isomorphism	+	appropriate functions, e.g. combinations of extract and concat
	missing entity	o	generation of values or default values for missing entities in source schema, not reasonable in each case loss of data for missing entities in target schema
	data type	o	conversion function in case of compatibility
	generalisation	o	appropriate functions, e.g. extract and concat, not always possible
	schematic discrep.	+	conversion function
	hierarchy	+	conversion with unnesting in case of mapping from structured to flat schema conversion with grouping in case of mapping from flat to structured schema
	Foreign key	+	generation of keys in case of foreign key in target schema conversion with join in case of foreign key in source schema
	constraints	o	no problem in case of weaker constraints in target schema, otherwise not resolvable
- : no support   o : conditional support   + : support			

### 3.1.5 Mapping Tools

There are some commercial schema mapping tools available. Among the tools analysed in [LN06] and applicable to XML schemata are Mapforce, Stylus Studio, Biztalk Mapper, Websphere Studio Application Developer, and the Clio research prototype. At the time of the analysis (published in 2006), all of these tools manifested limitations: *“We found that no tool performs well in all mapping situations and that many tools produce incorrect data transformations”* [LN06]. Despite these limitations, tools provide an indispensable support for schema mapping, even though the transformations they generate may have to be adjusted manually. Moreover, new versions are available at least for some of the commercial tools that may have improved with respect to the limitations detected.

## 3.2 Schema Integration

Schema integration is used when data exchange is based on more than three different, heterogeneous schemata and point-to-point mapping between each pair of schemata shall be avoided.

The general purpose of schema integration is to provide an intermediary for the data exchange between systems using different schemata. In such cases, data to be exchanged is then first

transformed from the source format to the intermediate format and then to the target format of the receiver of the message using the mapping techniques described in the previous subsection.

Schema integration comprises methods and techniques for integrating several different schemata into one integrated schema that logically subsumes the source schemata. Such an integrated schema serves as an intermediary for the data exchange between systems using different schemata where the point-to-point mapping between a source and a target schema and the related data transformation are replaced by a mapping from the source schema to the integrated schema, a mapping from the integrated schema to the target schema, and related data transformations. Obviously, if a point-to-point mapping between two schemata is not possible, a corresponding mapping via an integrated schema is also not possible.

[BLN86], one of the standard papers in the field, distinguishes four steps for integrating several schemata into one schema:

1. Pre-integration: An analysis is carried out to determine the schemata to be integrated, the order of integration, and the number of schemata to be integrated at one time.
2. Comparison of the schemata: Schemata are analyzed to determine the correspondences among concepts and to detect possible conflicts.
3. Conforming the schemata: Conflicts are resolved in close interaction with designers and users.
4. Merging and restructuring: The schemata are combined and possibly restructured in order to achieve the quality goals.

A distinction can be made between step-wise integration, where two schemata are integrated in each step, and n-ary integration, where more than two schemata are integrated in each step.

Desirable quality goals for the integrated schema are:

- *“Completeness and Correctness. The integrated schema must contain all concepts present in any component schema correctly. The integrated schema must be a representation of the union of the application domains associated with the schemas.*
- *Minimality. If the same concept is represented in more than one component schema, it must be represented only once in the integrated schema.*
- *Understandability. The integrated schema should be easy to understand for the designer and the end user. This implies that among the several possible representations of results of integration allowed by a data model, the one that is (qualitatively) the most understandable should be chosen”* [BLN86].

As one prominent example of schema integration methodologies, the approach of [SPD92] is briefly presented here. This approach is independent of specific data models, i.e. it can be applied to object-oriented model diagrams as well as to ER diagrams (the XML model did not exist yet at the time the approach has been developed).

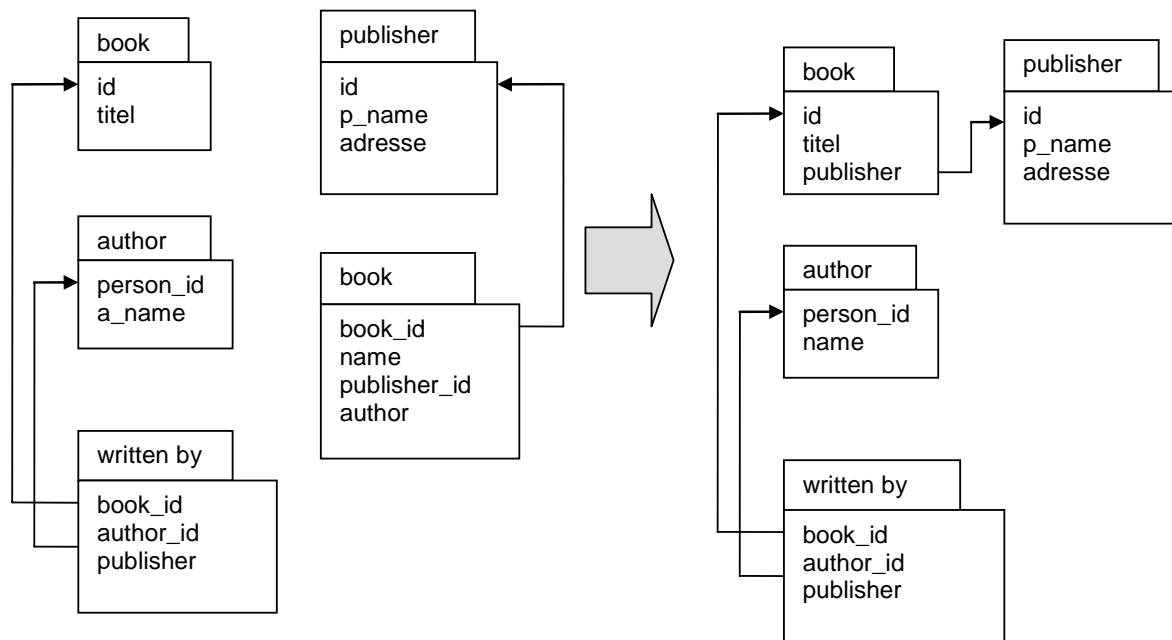
The approach is based on correspondence assertions between elements of the schemata to be integrated. These correspondence assertions correspond to the value correspondences introduced in subsection 3.1. Two elements are in equivalent correspondence if they activate/denominate the same concept. Other correspondences are subsumption, overlapping, and disjunction (see section 2.1).

In case of the object-oriented model, the five integration rules of the approach concerning the integration of two schemata (being presented here informally, for a formal definition see [SPD92]) are

- elements without correspondence in the other schema, i.e. occurring in only one of the schemata, are included in the integrated schema
- equivalent elements are included in the integrated schema together with the union of their attributes
- direct relations between equivalent elements are included in the integrated schema

- equivalent elements that are related by a path, i.e. a chain of elements related by relations, are included in the integrated schema
- equivalences between classes and attributes are included in the integrated schema in form of relations

The following example, adapted from [LN07], illustrates an integration of two schemata.



*Figure 13: Integration of two schemata*

In the example, there is no connection between the publisher of the “written by” relation and the id in the publisher relation due to the fact that the publisher in the book relation and in the “written by” relation could be different. For example, the publisher in one relation could be the one who first published the book, whereas the rights to the book could have changed, and the publisher in the other relation could be the one who actually holds the rights. In this case, the different publishers would denominate different concepts, i.e. the actual publisher versus the initial publisher.

### 3.2.1 Purposes

Mappings via an integrated schema face the same semantic conflicts as point-to-point mappings between a source and a target schema. Therefore, concerning the resolving of semantic conflicts, all of the schema mapping techniques described in subsection 3.1 can be applied.

general
The general purpose of schema integration is to provide an intermediary for the data exchange between systems using different schemata. Data to be exchanged is first transformed from the export format to the intermediate format and then to the import format of the receiver of the message using the mapping techniques described in the previous



2009-06-02

subsection.			
conflict-related			
conflict type		support	
data level	synonyms	+	transformation based on glossaries
	homonyms	-	not resolvable
	data representation	+	conversion function
	data unit	+	conversion function
	data precision	o	conversion function, conversion from lower to higher precision is not possible
	data language	+	translation based on controlled vocabularies
schema level	synonyms	+	building value correspondences
	homonyms	+	no correspondence
	hypernyms	o	appropriate functions, e.g. extract and concat, not always possible (see generalisation)
	overlapping	o	generally no correspondence (see 2.2.1), as no semantic equivalence
	schema isomorphism	+	appropriate functions, e.g. combinations of extract and concat
	missing entity	o	generation of values or default values for missing entities in source schema, not reasonable in each case loss of data for missing entities in target schema
	data type	o	conversion function in case of compatibility
	generalisation	o	appropriate functions, e.g. extract and concat, not always possible
	schematic discrep.	+	conversion function
	hierarchy	+	conversion with unnesting in case of mapping from structured to flat schema conversion with grouping in case of mapping from flat to structured schema
	Foreign key	+	generation of keys in case of foreign key in target schema conversion with join in case of foreign key in source schema
	constraints	o	no problem in case of weaker constraints in target schema, otherwise not resolvable
- : no support   o : conditional support   + : support			

### 3.2.2 Schema integration tools

Except research prototypes, no tools explicitly supporting schema integration are available, but mapping tools can be used at least for defining the mappings to and from the integrated schema.

Due to [LN07], though subject of research for more than thirty years, “*schema integration should rather be considered as an art than as a systematic activity*”.

## 3.3 Schema Matching

Schema matching is used as a supporting methodology for schema mapping under the following conditions

- huge schemata with identical identifiers, e.g. name and id, that due to their size cannot be represented graphically in an appropriate way
- complexity of the schemata to be mapped, e.g. deep hierarchies of XML schemata
- completely unknown and undocumented schemata
- high degree of heterogeneity of the schemata involved
- multilingualism
- cryptic identifiers of schema elements, e.g. caused by an omission of vowels due to conventional restrictions for the length of identifiers

The general purpose of schema matching is to detect proposals for correspondences between different schemata automatically. A matching tool analyses schemata to offer proposals for correspondences that have to be verified and accepted or rejected by a domain expert [RB01].

Schema matching is a support activity for schema mapping as well as for schema integration.

A classification of different matching approaches is given in [RB01]. In particular, there is a distinction between schema-based matching and instance-based approaches.

### 3.3.1 Schema-based matching

Schema-based matching is only applied to the identifiers and the structure of the schemata to be mapped. [RB01] distinguishes between the following aspects of schema based matching

#### LINGUISTIC APPROACHES

Linguistic approaches use names or descriptions to detect possible matches. In the case of XML schemata, name matching compares names of elements or attributes to find matches. “*Similarity of names can be defined and measured in various ways, including:*

- *Equality of names. An important subcase is the equality of names from the same XML namespace, since this ensures that the same names indeed bear the same semantics.*
- *Equality of canonical name representations after stemming and other preprocessing. This is important to deal with special prefix/suffix symbols (e.g., CName → customer name, and EmpNO → employee number)*
- *Equality of synonyms. (E.g., car ≡ automobile and make ≡ brand)*
- *Equality of hypernyms. (E.g., book is-a publication and article is-a publication imply book ≡ publication, article ≡ publication, and book ≡ article)*

- *Similarity of names based on common substrings, edit distance, pronunciation, soundex (an encoding of names based on how they sound rather than how they are spelled), etc. (E.g., representedBy  $\equiv$  representative, ShipTo  $\equiv$  Ship2)*
- *User-provided name matches. (E.g., reportsTo  $\equiv$  manager, issue  $\equiv$  bug)” [RB01]*

The edit distance is the minimum number of operations required to transform one string into another, where these operations are “insert”, “delete”, “replace”, and “match” of single characters (see [LN07]).

Description matching compares comments within schemata to detect equivalences between schema entities, e.g. “empn // employee name” in a source schema is likely equivalent to “name // name of employee” in a target schema.

#### GRANULARITY OF MATCH

Whereas element-level matching compares entities isolated from their embedding into a structure, e.g. element and attribute names of a source XML schema with element and attribute names of a target XML schema, structure-level matching compares combinations of elements appearing together in a structure. In the second case, even a partial match of the entities within a structure may be an indication for a match of the superior entity, e.g. a partial match of the attributes of two XML elements may indicate a match of these elements.

#### MATCH CARDINALITY

One or more elements of the source schema may match one or more elements in a target schema locally or globally. In a local match, “name” in a source schema matches “first name” and “last name” and therefore has local match cardinality 1:N. In a global match, “price” in a source schema is a (possible) match for “amount” and “cost” in a target schema and therefore has global match cardinality 1:N.

#### CONSTRAINT-BASED APPROACHES

*“Schemas often contain constraints to define data types and value ranges, uniqueness, optionality, relationship types and cardinalities, etc. If both input schemas contain such information, it can be used by a matcher to determine the similarity of schema elements. For example, similarity can be based on the equivalence of data types and domains, of key characteristics (e.g., unique, primary, foreign), of relationship cardinality (e.g., 1:1 relationships)” [RB01].*

#### REUSING SCHEMA AND MAPPING INFORMATION

Information of previous matchings may be exploited to detect new matchings, in particular for structures or substructures that often repeat, e.g. addresses. But finding the related substructure corresponding to a mapping is a match problem in itself. Generally, mappings should be handled with care when they stem from a different domain.

### 3.3.2 Instance based matching

Instance based matching is particularly used when the documentation of the schemata involved is limited [RB01]. Instance based matching inspects the data to derive schema-level matches from similarities of the related data. A precondition for applying instance based matching is that instance data are accessible.

Information retrieval mechanisms may be preferred for text elements, whereas for structured data as numerical or string elements, constrained-based characterisations, e.g. value ranges or character patterns can be applied to identify phone numbers, postal codes, etc.

### 3.3.3 Purposes

general			
The general purpose of schema matching is to detect proposals for correspondences between different schemata automatically. Schema matching is a supporting activity for schema mapping as well as for schema integration.			
conflict-related			
conflict kind		support	
data level	synonyms	+	detection based on glossaries
	homonyms	+	detection based on glossaries
	data representation	+	documentation and instance matching
	data unit	+	documentation matching
	data precision	+	documentation and instance matching
	data language	+	instance matching, but should be obvious
schema level	synonyms	+	detection of possible correspondences by each of the techniques introduced above
	homonyms	+	detection based on instance matching or documentation
	hypernyms	+	detection based on controlled vocabularies and instance matching
	overlapping	o	possibly detected by instance matching
	schema isomorphism	-	
	missing entity	-	
	data type	+	
	generalisation	o	detection based on controlled vocabularies and instance matching
	schematic discrepancies	-	
	hierarchy	-	
	Foreign key	-	
	constraints	+	
- : no support   o : conditional support   + : support			

### 3.3.4 Schema Matching Tools

There are prototype schema matching tools available, each of them combining several matching techniques, but apart from matching functionality implemented in schema mapping tools, no commercial matching tools are available.

### 3.4 Data Transformation

Data transformation is achieved by applying the transformation instructions generated in the last step of the mapping process. The main problems occurring in the data transformation process, not classified as semantic conflicts in the literature, are

- missing values
- wrong values, e.g. typing errors that impede correct translations
- wrong foreign key/key relations
- abbreviations
- diverted attributes, i.e. values for an attribute that has been “forgotten” in a schema, are added to another attribute
- duplicates.

Duplicates are a severe problem in the area of federated databases but are considered to be of minor relevance for data exchange in the SEMIC.EU context. The other problems should be avoided as far as possible on the schema level, e.g. by defining appropriate constraints to avoid missing values or by restricting the values through controlled vocabularies to avoid wrong values and abbreviations. Wrong foreign key/key-relations cannot usually be detected automatically.

The methodologies described in this section dealt with detecting and resolving semantic conflicts. These conflicts result from the fact that different schemata use a different naming and different structures to capture the same information. To detect and resolve these conflicts, semantically equivalent structures within different schemata have to be detected, in particular, by identifying relationships between names as synonyms or narrower/broader term-relationships.

Controlled vocabularies are the means to represent such relationships. They aim at the precise determination of the meaning of terms and the relationship between them. The following section will present the different types of controlled vocabularies in detail, focussing on their support for detecting semantic conflicts.

## 4. CONTROLLED VOCABULARIES

The strong need to introduce vocabularies is determined by the fact that there are many areas where it is necessary to have a common, unambiguous understanding. A common understanding is usually considered to be a prerequisite for precise work processes where collaborating partners are required to communicate effectively and efficiently in order to produce consistently correct results. In order to ensure consistency and a high quality of the applied vocabulary, it must be controlled according to [Pry99, 7th edition, p.163], where a controlled vocabulary is defined as “*a specified list of terms with a fixed and unalterable meaning (...). The control is intended to avoid the scattering of related subjects under different headings (...). The list may be altered or extended only by the publisher or issuing agency*”.

Due to the fact that the aim of controlled vocabularies is the unique determination of the meaning of terms, they play a crucial role in conjunction with semantic interoperability. In the concrete context of SEMIC.EU, controlled vocabularies are of significant importance in their function as components in the process of semantic disambiguation of pan-European information models and semantics-preserving data exchange according to [FI08]. Semantic interoperability may only be guaranteed if semantic conflicts, which are described in section 2, are handled in a way that transformations between different data entities fully preserve the initial semantics. Controlled vocabularies are considered as a fundamental methodology for addressing and handling semantic conflicts.

The actual extent to which a controlled vocabulary may handle semantic conflicts depends on its type. Therefore, the most prominent vocabulary types that have been invented in the last few decades are introduced in the following sections.

A vocabulary type is strongly interrelated with its expressiveness, analogous to other disciplines of computer science, e.g. programming languages or information modelling. Essentially, the level of expressiveness depends on the various potential options that may be used to interrelate terms of the vocabulary and to state logical axioms over the term set.

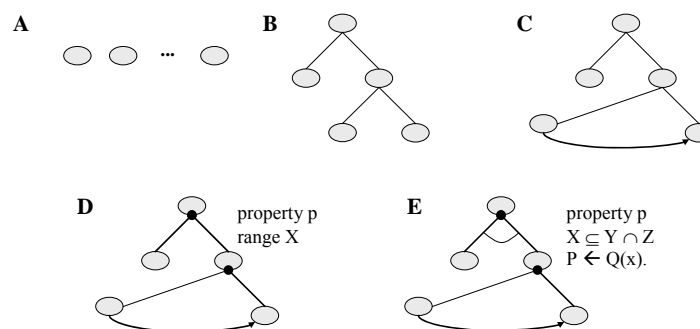


Figure 14: Types of controlled vocabularies

The most common types of controlled vocabularies are:

- A – glossaries (dictionaries), code lists,
- B – taxonomies,
- C – thesauri,
- D – ontologies, and
- E – full-fledged ontologies.

Figure 14 provides a short overview of the vocabulary types identified above and illustrates their structure and interrelation of terms as well as their ability to include conceptual modelling constructs or logical axioms (ellipses indicate terms and arcs depict interrelations). The gradual progression of expressiveness may be reflected by:  $A \subset B \subset C \subset D \subset E$ . A notable point here is that any means that may be used to build expressions at one level is also included and fully usable at the next higher levels. Therefore, the concrete features of one level are fully inherited by the next higher one, so that features are in fact propagated through the above chain of vocabulary types.

Besides a description of and introduction to the different vocabulary types themselves, the following sections will also discuss the fundamental purposes of a controlled vocabulary with a particular focus on their potential in handling semantic conflicts. Common data conflicts as well as the schema conflicts described in section 2 will be related to each vocabulary type.

However, due to the manifold and complex nature of potential semantic conflicts, controlled vocabularies may not represent a feasible approach to solving all and each type of the problems mentioned. In particular, the handling of some data conflicts (e.g. conversion of data formats), or structure-oriented schema conflicts (e.g. conflicts concerning technical entities like database keys) may be well beyond the means of controlled vocabularies.

Section 4.1 introduces the vocabulary types in more detail, whereas section 4.2 describes the currently used common languages for controlled vocabularies and the concrete extent to which they adequately represent a vocabulary type. The last section, 4.3, deals with the development process for ontologies as the most expressive vocabulary type.

Generally, this section focuses on the use of controlled vocabularies in conjunction with semantic conflicts, which is one of the key methods in the area of semantic interoperability and information integration. However, since semantic conflicts between controlled vocabularies themselves and the exchange / transformation of vocabularies between collaborating partners is a very active area of research and requires a rather deep level of consideration, these might not be fully addressed within this study.

## **4.1 Types of Controlled Vocabulary**

Based on the above classification (see Figure 14), this section describes each vocabulary type in greater detail. After a short introduction of the purposes of the vocabulary type in general and with respect to semantic conflicts, the detailed characteristics of each type are described. At the end of each subsection, a table summarises the purposes of the vocabulary type. Finally, section 4.1.6 summarises all findings and additionally illustrates the relationship between conflicts and vocabulary types.

### **4.1.1 Glossaries and Code Lists**

The general purpose of glossaries and code lists is to provide a common understanding of the terms (and codes) within the currently applicable domain. Obviously, no structure is predetermined for code lists, but the ability to represent synonyms and homonyms clearly supports the detection and resolution of conflicts related to these constructs. Synonym lists (also called synonym rings) can be used to determine the semantic equality of terms when translating data from source to target. Furthermore, detecting different names for schema elements that denote the same concept can be supported by inspecting the synonym ring.

Another purpose of glossaries is to provide support for detecting homonymy conflicts whenever a qualification construct for representing homonyms is provided. For instance, if two schemata use the same term at different locations, these can be represented by surrounding names (context). In order to detect such homonymy, one can consult a glossary with the context as a parameter so that different

qualifiers can be found. However, if a potential qualification is only considered in the descriptive/explanatory text of a glossary, only manual detection support is provided.

Clearly, glossaries and code lists may help to avoid semantic conflicts because different schema developers are supported in using the same terms for schema elements.

#### CHARACTERISTICS

Glossaries and dictionaries, also known as *flat term lists*, enlist terms with a unique description. It is the simplest form of a controlled vocabulary, where no predetermined structure is applied to the term set. Whereas traditional dictionaries usually refer to rather domain-independent term lists where sometimes also grammatical information is available, glossaries are used to describe terms from a specific domain. A domain-dependent glossary enlists the terms of a domain language in factual form so that an unambiguous usage is ensured.

Apart from the long tradition of using glossaries since the ancient world, glossaries in the age of computing are often related to artefacts stored on computers. In this case, these glossaries are also called data dictionaries. A data dictionary can be seen as a collection of identifying descriptions of items within a data model. Its purpose is to avoid semantic mismatches when using this data model for application development.

<b>FDDI</b>	Fiber Distributed Data Interface. An emerging standard for network technology based on fiber optics that has been established by the American National Standards Institute (ANSI). FDDI specifies a 100-million bit per second data rate.
<b>File</b>	A collection of information on a disk, usually a document or a program, that's lumped together and called by one name.
...	

*Figure 15: Glossary fragment*

Because of its simple structure, the skills for creating and maintaining a glossary are usually available among the members of a typical working group of computer professionals. However, without a clearly structured process of analysing the domain terminology, the quality of the glossary may be inadequate.

Glossaries are one of the widespread forms of controlled vocabularies. Almost every domain features its own specialised terminology, thus creating a broader need to define and employ glossaries as a simple means of achieving a common understanding. Also due to their simplicity, glossaries may be easily communicated through the World Wide Web. For example, many sites like [Gloss] offer a large number of glossaries, ranging from 'Arts & Culture' and 'Science' to 'Transport'. One typical and exemplary glossary fragment from [Gloss], which are normally ordered alphabetically, is shown in Figure 15. Whenever an underlying structure is of a relatively simple nature, usually no major staff efforts are required to develop a representation format.

Essentially, glossary items are tuples of the form:

- $(term, [term_1, \dots, term_n], description)^1$ .

The creator of the glossary may assign the term to be the *representative* for the equivalence class given by  $[term, term_1, \dots, term_n]$ . The representative is usually stated at the first position of the tuple.

<sup>1</sup> the term list  $term_1, \dots, term_n$  states the synonym ring (equivalence class)



In the case of code lists, the set of terms may well be of different nature in comparison to glossaries. While traditional glossaries focus on the clarification of terms stemming from a domain-specific terminology; terms in the case of code lists can also include concrete objects. For instance, schools in administrative districts could have the codes S1 to SN assigned, where every code represents the name of a particular school. More generally, code lists are lists of tuples of the form:

- $(code, term, [term_1, \dots, term_n], description),$

where the set of terms and the set of codes correlate to each other in a bijective relationship. However, the provision of terms is optional; many code lists only contain the code and the description.

Similar to the idea of uniform resource identifiers in the World Wide Web, there are many code lists that extend the forms introduced so far. A common extension is to structure code lists in a way that they can serve as object pools. Clearly, the form above, consisting of a code, a term, and its description, may already be considered to be an object. In this case, the code serves as the object identifier (OID) for the object containing an attribute for the term and an attribute for its description.

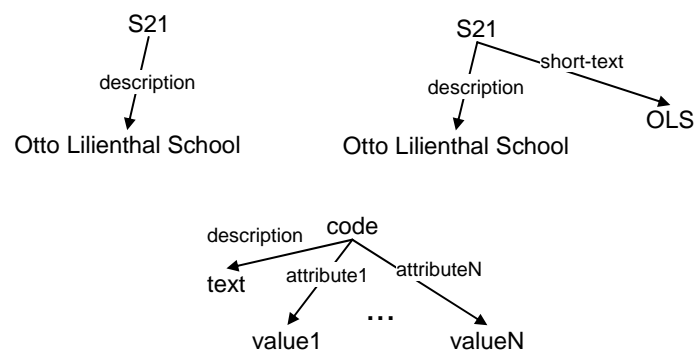


Figure 16: Code lists

More elaborated structures are used in the area of the public administration. For example, a code list for schools in an administrative district normally introduces a short descriptive text besides the long description. Another example is to incorporate common school attributes into the code list, such as the building date of the school, which might be of interest to administrative sections dealing with construction services. These attributes represent *configuration attributes* that are used to initialise the frontend of applications in the area of e-government. Figure 16 shows an example in a tree representation.

The only condition that should be fulfilled concerning the structure of code lists is the *flatness condition*, i.e. attributes must be simple values that may not be codes themselves.

Some kinds of glossaries support the usage of homonyms. To incorporate homonyms within the glossary, terms are qualified by the scopes. Because homonym handling is at least mandatory for thesauri, see section 4.1.3 for details.

PURPOSES

general
The aim of glossaries and code lists is to establish a common understanding of terms without introducing a predetermined structure or stating their relation. In the context of semantic interoperability, its function is to avoid the use of different terms for the same concept by including synonym rings.

conflict-related			
conflict type		support	
data level	synonyms	+	handling by synonym rings
	homonyms	o	if scope/qualification is allowed
	data language	+	handling based on language tags
schema level	synonyms	+	handling by synonym rings
	homonyms	o	if scope/qualification is allowed
	hypernyms	-	
	overlapping	-	
	schema isomorphism	-	
	generalisation	-	
- : no support   o : conditional support   + : support			

#### 4.1.2 Taxonomies

The general purpose of introducing taxonomies is to provide users with the additional ability to define hierarchical structures over the terms. This, for example, can support more elaborate term browsing in frontends. But apart from this general purpose, taxonomies can support the handling of a class of semantic conflicts that cannot be covered by glossaries. Essentially, every conflict that concerns subsumption of terms can be covered, namely hypernym and generalisation conflicts (see section 2.2.1 and 2.2.5).

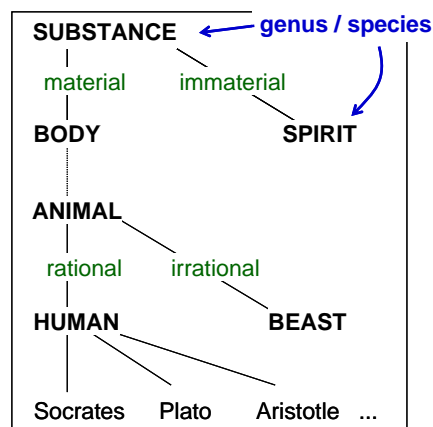


Figure 17: Tree of Porphyry

Detecting hypernymic relationships between values and schema elements is supported by inspecting the taxonomic hierarchy. A conflict occurs if, for instance, one schema uses the specific term while another schema uses the more general term. In contrast to glossaries, with taxonomies it is possible to determine basic concept similarity by calculating the taxonomic distance (path length between terms in the hierarchy).

## CHARACTERISTICS

So far, glossaries can only represent flat-term structures. The next level of structural complexity is introduced through taxonomies. As seen in Figure 14, the transition from term lists to taxonomies is indicated by the introduction of a tree structure, i.e. terms within taxonomies are structured in a hierarchical manner. Similar to glossaries, taxonomies have been used since the ancient world. One of the first taxonomic structures, shown in Figure 17, stems from Porphyry (around 3rd century BC), which classifies substances of the world in hierarchical form (for detailed discussions see [Sow99]).

In taxonomies, terms are set in a parent-child relationship. In contrast to a rigid interpretation of this hierarchical relationship with a formal subset relationship, which also holds for relationships in Figure 17, taxonomies allow for setting two terms  $s$  and  $t$  in a subsumption relationship, where  $s$  is more general and  $t$  more specific. Various kinds of taxonomic relationships are known:

- broader / narrower
- part / whole
- genus / species
- class / subclass
- class / instance

Whereas the first two relationships are of a more non-formal nature, the last three are interpreted formally. As usual, these relationships are interpreted set-theoretically, assuming that every term in the taxonomy stands for a set of things. If for two terms  $s$  and  $t$  the relationship  $R$  is stated, where  $R$  stands for genus / species or class / subclass, then the set of things denoted by  $t$  is a subset of the things denoted by  $s$ . For example, if

- *vehicle* subclass *sports-car*

is stated within the taxonomy, then the set of all sports cars is contained in the set of all vehicles. Moreover, a set-theoretical interpretation of terms and their interrelations yields a proper handling of the class / instance relationship. More precisely, if terms  $s$  and  $t$  are in the relation class / instance, then the instance denoted by  $t$  is an element of the set of things denoted by  $s$ . For example, in Figure 17, Aristotle as a unique person, is in the class / instance relation to human. Another example is the definition

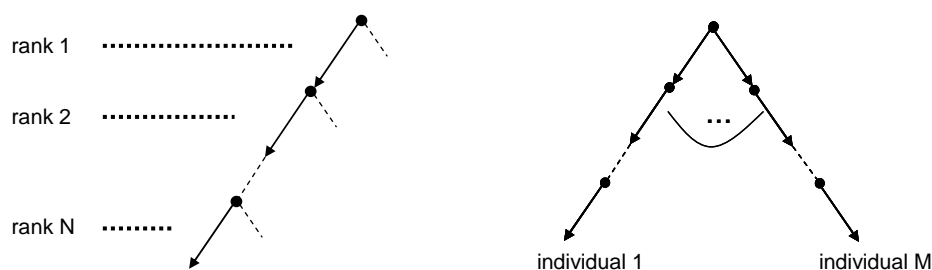
- *sports-car* instance *porsche-BE36543*

Hence the specific car *Porsche-BE36543*, which is a unique object in the world, is an element of the set of all sports cars. Inclusion of instances allows for representing concrete objects of the world within taxonomies. It must be noted that the formal relationships described so far are redundant in the sense that the class / instance relationship can be simulated with the class / subclass relationship if we represent an instance by a singleton set containing this instance.

Taxonomies do not enforce fulfilment of the formal definition by the hierarchical relationship. For instance, let the relation

- *recreation* narrower *bathing-fashion*

be given within the classification schema of a shopping portal. Underlying the set-theoretic framework, it is not intuitive that *recreation* denotes a set of elements where *bathing-fashion* is contained, but it is an acceptable classification for browsing a shop portal. Also, usual folders of a directory structure can be seen as a taxonomy that serves as a classification schema for documents but do not have to follow a rigid formalisation.



**Figure 18: Taxonomic rank**

Taxonomies do not have to form a strict tree. This fact can be directly derived from the fact that all mentioned set relationships are not restricted to the constraint that children must have a single parent. If more than one parent exists for a child, we also speak of poly-hierarchical taxonomies. Clearly, things denoted by terms can be subsumed by more than one term. This mechanism is analogous to the mechanism of multiple inheritance used in object-oriented programming. For instance, the computer language C++ can be subsumed by the term object-oriented language as well as by the term hardware-oriented language, considering that C as a subset of C++ is used for programming hardware units.

For some taxonomies, the degree of freedom concerning the depth and structure of the taxonomy tree is restricted. The depth of a taxonomy is determined by the set of allowed taxonomic ranks where each rank can be seen as a relationship name for the parent-child relationship. Figure 18 shows the general form of fixed-rank taxonomies. The depth of the taxonomy is given by the most specific rank N. Moreover, fixed-rank taxonomies are balanced in the sense that every individual has to be classified with rank 1 up to rank N, i.e. the taxonomic path from the taxonomic root to each individual must have the length N.

The most famous taxonomy with a fixed set of ranks is the Linnaean taxonomy [Atr90]. The potency of the taxonomy developed by Linnaean is its usability to create a simple and convenient system for organizing the different sorts of organisms. For every species, a unique and stable name is defined.

So far we have considered flat lists of terms and hierarchies of terms that are related to glossaries/code lists and taxonomies, respectively (see Figure 14, A and B). Whereas A is more-or-less structure free, taxonomies are able to represent the subsumption structure over the term set. One can speak of a first step of representing semantic information that goes beyond a simple semantic description of single terms. Using this semantic information in information retrieval increases the search quality in a fundamental way. For example, if the simple taxonomy fragment

- *animal* subclass *dog*, *cat*, ...

were used in a search, declaring *animal* as a keyword could also retrieve documents where *dog* and *cat* are contained, of course with a lower weight than documents that contain *animal*.

#### PURPOSES

general			
Taxonomies extend glossaries by introducing hierarchical relationships between terms. Hierarchical relationships allow for richer navigational support in conjunction with concept browsing as well as increase search quality for information retrieval systems.			
conflict-related			
conflict type		support	
data	synonyms	+	improved synonym detection by determining

level			taxonomic distance
	homonyms	o	→ <i>Glossaries</i>
	data language	+	→ <i>Glossaries</i>
schema level	synonyms	+	improved synonym detection by determining taxonomic distance
	homonyms	o	→ <i>Glossaries</i>
	hypernyms	+	detecting hypernyms by means of taxonomic hierarchy
	overlapping	-	
	schema isomorphism	-	
	generalisation	+	detecting hypernyms by means of taxonomic hierarchy
- : no support   o : conditional support   + : support			

#### 4.1.3 Thesauri

Whereas taxonomies are only able to represent hierarchical term structures, a thesaurus is the first vocabulary type that introduces associative relationships.<sup>2</sup> As a result, the semantic similarity of values and schema names, which is used to identify possible synonyms in the context of conflict handling, can be determined by traversing these associative relationships.

The detection of synonymy candidates is much more precise compared to glossaries and taxonomies. Again, navigational support is enriched, and the search quality of IR systems is enhanced by incorporating semantically related terms. Furthermore, a thesaurus is the first vocabulary type that has an *explicit* support for the representation of homonyms, which definitely makes handling homonym conflicts possible.

##### CHARACTERISTICS

According to [ANS05], “*A thesaurus is a controlled vocabulary arranged in a known order and structured so that the various relationships among terms are displayed clearly and identified by standardized relationship indicators. Relationship indicators should be employed reciprocally*”. Whereas relationships between terms are not considered for glossaries, thesauri weaken the restriction of taxonomies to use only the subsumption relationship and to allow the use of term relationships from a fixed-relationship set.

From a structure point of view, the next step towards richer semantic information for term structures is reflected by the transition from B to C in Figure 14, namely the step from tree structures to net (or graph) structures. More precisely, B can be seen as an acyclic directed graph, whereas C allows for cyclic structures. Structurewise, thesauri introduce associative relationships that do not have to follow hierarchical constraints.

The relationship set (also called the set of relationship indicators) can be divided into the following types

- Equivalency relationships

<sup>2</sup> Note that the term thesaurus used in the computer science community differs from more popular interpretations where this term only stands for a synonym lexicon.

- Hierarchical relationships
- Associative relationships

Table 2 shows a refinement of these relationship types according to [ANS05]. Of course, glossaries and taxonomies are restricted versions of thesauri because (extended) glossaries only allow statements of synonymy; taxonomies, furthermore, make use of hierarchical relationships (generic or instance).

Relationship Type	Example
<b>Equivalency</b>	
Synonymy	UN / United Nations
Orthographic variants	pediatrics / paediatrics
Near synonymy	sea water / salt water
<b>Hierarchy</b>	
Generic or IsA	birds / parrots
Instance or IsA	sea / Mediterranean Sea
Whole / Part	brain / brain stem
<b>Associative</b>	
Cause / Effect	accident / injury
Action / Product	writing / publication
Action / Property	communication / communication skills
Action / Target	teaching / student
Raw material / Product	grapes / wine
Discipline or Field / Object or Practitioner	neonatology / infant

*Table 2: Thesaurus relationship types (acc. to [ANS05])*

Furthermore, thesauri take special care of terms from the linguistic perspective that is important to handle synonym and homonym conflicts. Due to more expressivity by associative relationships, thesauri have more precision in detecting synonym candidates during conflict detection.

Apart from synonymy, which is treated as a special equivalency relationship, homonymy is handled with a special construct called ‘scope’. Homonymy occurs when one term has multiple meanings. Usually scopes are stated with qualifiers that disambiguate the term. For example, the term ‘seals’ can be disambiguated by stating the following qualifiers:

- seals (animals)
- seals (law)
- seals (numismatics)

2009-06-02

## PURPOSES

general			
Thesauri extend taxonomies by introducing explicit constructs for homonyms and associative relationships. Associative relationships increase the expressiveness by relating terms according to their semantic proximity. Again, navigational support is enriched and the search quality of IR systems is increased by incorporating semantically related terms.			
conflict-related			
conflict type		support	
data level	synonyms	+	improved conflict handling by means of associative relationships
	homonyms	+	explicit representation of homonyms by means of the qualification/scope construct
	data language	+	→ <i>Taxonomies</i>
schema level	synonyms	+	improved conflict handling by means of associative relationships
	homonyms	+	explicit representation of homonyms by means of the qualification/scope construct
	hypernyms	+	→ <i>Taxonomies</i>
	overlapping	-	
	schema isomorphism	-	
	generalisation	+	→ <i>Taxonomies</i>
- : no support   o : conditional support   + : support			

## 4.1.4 Ontologies

Ontologies introduce another grade of freedom, namely the ability to define arbitrarily named relationships between terms instead of restricting them to a fixed relationship set to as it is done for thesauri (relationship indicators). Additionally, constructs for defining schemata are introduced that allow for the development of information models. Therefore, one of the general purposes of ontologies is a seamless integration of terminologies and information modelling, which offers the benefit that no technical exhausting bridges between models and terminologies have to be designed.

The purpose of ontologies related to semantic conflict handling is to cover classes of conflicts that cannot be covered by the vocabulary types discussed so far, namely schema isomorphism conflicts (see section 2.2.2). Here, ontologies can serve as reference models that define concepts together with their attributes. By looking up the attribute part of the ontology and comparing it with the schemata, a reference concept can be determined.

## CHARACTERISTICS

According to the original definition [Gru08], “an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members)”. In other words, ontologies can serve as conceptual models that define properties with their domain and ranges used by instances. For example, a property ‘age’ can be defined with the range

‘integer’ and domain ‘employee’, assuming the area of working life. In terms of structure, there is no difference from thesauri, i.e. the net representation remains valid.

Due to the formal underpinning of ontologies, which in general is hidden from the user or creator of an ontology, ontology tools and systems allow for performing ‘calculations’ over the term net, commonly known as reasoning. The first application of reasoning is to check the validity of the instance values and structure with respect to the schema part. One of the extended applications of reasoning is to deduce taxonomic information not stated explicitly within the ontology.

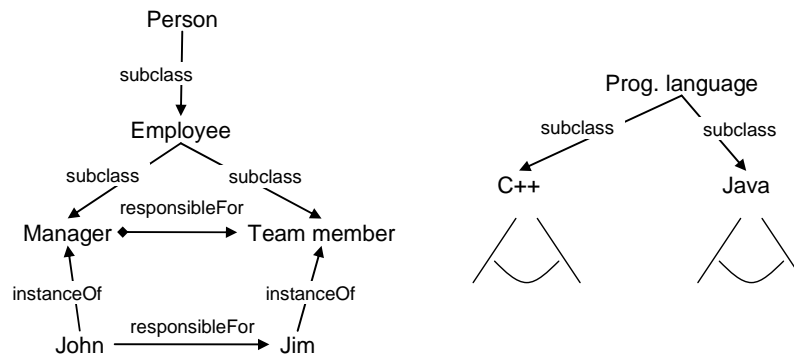


Figure 19: Ontologies

Figure 19 shows a sample ontology. In it, the usual term net from taxonomies and thesauri is extended by arcs between terms (mostly called classes or concepts within the related community) representing a property (or attribute) definition. For example, team members have responsible managers, and both are specialisations of ‘employees’, which, in turn, are specialisations of ‘people’. Based on this information, a reasoner will deduce that team members are people. More sophisticated applications of reasoning will be described in the next section.

Furthermore, a reasoner can directly check that the instance net at the bottom of the ontology is valid with respect to the upper part. At the right side of Figure 19, a taxonomy for programming languages is depicted, which could be used as skills in conjunction with the left part in the context of job portals. This exemplifies that one of the biggest advantages of ontologies is the unified approach to representing pure term structures as is done by thesauri, as well as conceptual models analogous to schema definitions.

#### PURPOSES

general			
Ontologies extend thesauri by incorporating schema elements and by the admission of arbitrarily named relationships. Schema elements and arbitrarily named relationships support development of information models. Precise underlying formalization supports facilitation of reasoners to deduce new conceptual relationships.			
conflict-related			
conflict type		support	
data level	synonyms	+	improved synonym handling by means of user-defined relationships
	homonyms	+	→ <i>Thesauri</i>
	data language	+	→ <i>Thesauri</i>



schema level	synonyms	+	improved synonym handling by means of user defined relationships
	homonyms	+	→ <i>Thesauri</i>
	hypernyms	+	improvement by involving user-defined relationships and reasoning
	overlapping	-	
	schema isomorphism	+	Ontology as a reference model wrt. properties/attributes
	generalisation	+	improvement by involving user-defined relationships and reasoning
- : no support   o : conditional support   + : support			

#### 4.1.5 Full-Fledged Ontologies

Full-fledged ontologies overcome the drawback of normal ontologies, which is that no complex concept expressions suitable to detect overlapping conflicts are provided (see section 2.2.1). These kinds of conflicts can only be handled by constructs that allow for stating complex expression over concepts. For instance, full-fledged ontologies are able to express the fact that two concepts are not disjointed. Exactly this information can be used to determine candidates for overlapping conflicts.

The integration of reasoners for full-fledged ontologies leads to another benefit. Hidden information, i.e. concept relationships not explicitly stated, can be derived by using formal reasoning. By using this facility, more accurate synonym detection in conjunction with conflict handling can be performed.

##### CHARACTERISTICS

All of the kinds of controlled vocabularies introduced so far can be expressed by means of simple factual statements that can be represented adequately by nets or expressions like

- *Employee* subclass *Manager*
- *Manager* responsibleFor *Employee*

As described in the last section, ontologies have strongly formalized semantics. This formal framework is essentially based on set theory, and it must be mentioned that this is not a unique property of ontologies but also a property of common conceptual modelling languages in the direction of entity-relationship modelling. Whereas conceptual modelling languages do not go beyond the level of the simple factual statements introduced so far, full-fledged ontologies make direct use of the set theoretical formalisation by offering the user more constructs, thus achieving a massive increase in expressivity.

Modern full-fledged ontologies are based on Description Logics, which are a highly developed branch within the artificial intelligence area. Nearly one decade of research has led to solid results that are applied in many fields of computer science [BCM+03]. Extensions that are offered can be divided into two kinds of constructs:

- concept expressions
- rules over concepts

Assuming that concepts denote sets of individuals, set constructs (operators and predicates) can be easily ‘lifted’ to developers of controlled vocabularies. Table 2 shows these operators, predicates, and related examples.

Construct	Example
<b>Operator</b>	
union ( <i>or</i> )	<i>Male or Female</i>
intersection ( <i>and</i> )	<i>Adult and Male</i>
complement ( <i>not</i> )	<i>Male and not Adult</i>
<b>Predicate</b>	
equality	<i>MaleUnderage equals Male and not Adult</i>
inclusion ( <i>containedIn</i> )	<i>MaleBaby containedIn MaleUnderage</i>

*Table 3: Concept expressions*

A union operator applied to two concepts denotes the union of the sets denoted by each concept. Therefore the expression *Male or Female* denotes the set of all males and females together. *Adult and Male* stands for the concept that denotes all adult males.

*Male and not Adult* denotes underage males, from which the new equal concept *MaleUnderage* is constructed by means of the equality predicate. Every operator-constructed expression stands for a new concept. Clearly, the equality predicate and the inclusion predicate are equivalent to synonymy and the taxonomic relation already known from controlled vocabularies described in the last sections, but operators can be seen as one of the essential extensions of simple ontologies.

Another kind of operators is used in conjunction with properties, also called roles in Description Logics. Similar to attributes in object-oriented programming, properties are defined on the schema level and applied on the instance level. An example of this was shown in the left part of Figure 19. Note that property definitions are knowledge for its own sake and not only meaningful for constraining instances. For example, the statement

- *Employee worksFor Company*

is part of knowledge of professional life. Assuming a knowledge base including this information, one can state the query

- retrieve all relationships between persons and companies

whose evaluation returns the *worksFor*-relationship as well as, for instance, the *advises*-relationship. Of course, under the view of data modelling, the property definition above is part of schema definition in order to enforce that every employee is linked to the company he works for. Full-fledged ontologies provide sophisticated constructors for the properties shown in Table 4.

Construct	Example
universal restriction ( <i>all</i> )	<i>all child Human</i>
existential restriction ( <i>some</i> )	<i>some hobby Ballsports</i>
qualified cardinality restriction ( <i>at-least</i> )	<i>at-least 4 tire WinterTire</i>

qualified cardinality restriction (at-most)	<i>at-most 2 hobby Sports</i>
--	-------------------------------

Table 4: Property expressions

The universal restriction construct defines the range type of a property. For example, the expression *all child Human* stands for the set of individuals who have a human as a child, if they have a child at all. Clearly, non-humans with offspring are not denoted by this expression. Existential restrictions enforce the existence of individuals for a property. An example is the expression *some hobby Ballsports* that stands for a set of individuals who have a hobby of the type *Ballsports*. Cardinality restrictions have usual semantics, i.e. they enforce a specific number of relations an individual can have.

As depicted in Figure 14 (E), full-fledged ontologies allow the definition of rules over the concept space. Of course, due to the precise semantics of ontologies, rules can be used for standard ‘calculations’ hidden to the developer. One of the most important rules is the transitivity rule, which determines the correct subsumption relationships between concepts. Formulated in usual predicate logic notation, subsumption resolution can be defined by the rule

- $subclass(X,Y) \text{ and } subclass(Y,Z) \rightarrow subclass(X,Z)$

where  $X,Y,Z$  are variables to which arbitrary concepts can be bound. More sophisticated rules are necessary to enable more complex reasoning over the concept expressions a user has stated. For example, assuming the last two rows in Table 3 and transitivity, it can be deduced that male babies are not adults due to the fact that

- *MaleBaby containedIn (Male and not Adult)* and thereby *MaleBaby containedIn not Adult*

Systems without reasoning support would not be able to derive such self-evident facts unless they were directly included in the ontology. Apart from hidden standard rules, user-defined rules substantially increase the expressivity of ontologies. The most famous example is the uncle-relationship, which is not expressible without rules:

- $parent(X,Y) \text{ and } brother(Y,Z) \rightarrow uncle(X,Z)$

Assuming a family ontology with individuals for which the *parent* and *brother* properties are assigned, the *uncle* property can easily be derived.

It has to be emphasized that concept expressions and rules are additional features that allow for the inclusion of expressive facts about the domain in our controlled vocabulary. Whereas controlled vocabularies without rules force the developer to state all knowledge explicitly, full-fledged ontologies enable the derivation of new facts from the given facts. This distinction is directly related to two kinds of knowledge representation, namely intensional and extensional knowledge, which is exemplified in Figure 20.

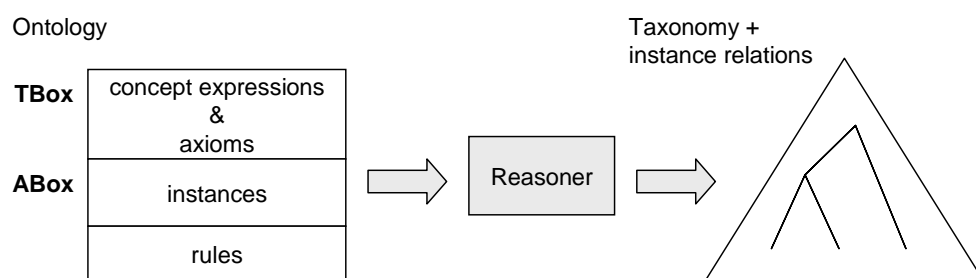


Figure 20: Intensional and extensional knowledge

On the right side, a full-fledged ontology is shown with its three basic components, namely concept expressions and axioms, instances, and rules. The component where concept expressions and axioms are located is usually called the terminological box (TBox); instances are defined in the assertional box (ABox).












On the left side, Figure 20 shows a full-fledged ontology stating intensional knowledge: here, the taxonomy is shown that represents extensional knowledge derived from the right part. Clearly, all controlled vocabularies considered so far can also be expressed in full-fledged ontologies.

#### PURPOSES

general			
Full-fledged ontologies extend ontologies by allowing complex concept expressions, property expressions, and rules over the concepts space. These elements significantly increase ontologies' expressiveness. Thereby, support for intensionally expressing sophisticated relationships between concepts is provided that makes the deduction of new conceptual structures by means of reasoners possible. Also, rules support the intensional definition of relationships based on logic formalism.			
conflict-related			
conflict type		support	
data level	synonyms	+	improved synonym handling by formal equivalence axioms
	homonyms	+	→ <i>Ontologies</i>
	data language	+	→ <i>Ontologies</i>
schema level	synonyms	+	improved synonym handling by formal equivalence axioms
	homonyms	+	→ <i>Ontologies</i>
	hypernyms	+	improvement by reasoning over concept expressions
	overlapping	+	overlapping detection by means of concept expressions
	schema isomorphism	+	→ <i>Ontologies</i>
	generalisation	+	improvement by reasoning over concept expressions
- : no support   o : conditional support   + : support			

#### 4.1.6 Summary of Purposes

In the context of SEMIC.EU, which has its focus on interoperability, controlled vocabularies are one of the main artefacts to ensure common understanding of assets as a prerequisite for building interoperable applications on a pan-European level. In particular, controlled vocabularies and ontologies - being the most elaborate - have to be used to handle semantic interoperability conflicts. The following table summarizes the purposes of controlled vocabularies in regard to conflict handling that were given at the end of each section describing vocabulary types.

conflict type		support from				
		Full-fledged ontologies	Ontologies	Thesauri	Taxonomies	Glossaries
data level	synonyms					
	homonyms					
	data language					
schema level	synonyms					
	homonyms					
	hypernyms					
	overlapping					
	schema isomorphism					
	generalisation					

It follows from the above table that classical naming conflicts on the data and schema level, namely synonyms and homonyms (if homonymy constructs are available), can be handled by all vocabulary types. Data language issues, too, can be covered starting from glossaries. A vocabulary of the type *taxonomy* is the first level where conflicts relevant in conjunction with hypernymic structures can be handled.

Ontologies introduce precise semantics and a wide range of constructs describing information models and conceptual knowledge. Based on these constructs and enabled by reasoning techniques, ontologies increase quality when dealing with general naming conflicts and conflicts related to hypernymic relationships. Moreover, schema elements in ontologies support handling of isomorphism conflicts. Only full-fledged ontologies are able to support overlapping and disjunction conflicts due to their facility to express complex concept expressions.

## 4.2 Standard Languages for Controlled Vocabularies

Due to the nature of SEMIC.EU, controlled vocabularies should be specified by means of standard languages. Standard languages have the clear advantage that they enable potential developers all over the world to contribute in an organized way. The more complex a language for controlled vocabularies is, the more benefits are given by relying on a standard because the efforts for developing language-dependent components like parsers, as well as tools for creating and visualizing controlled vocabularies, are shared within the community.

Of course, if the chosen type of vocabulary is simple, it seems to be sufficient to develop a separate representation format for the vocabulary without expending the effort for extensive investigation about standards. For instance, simple glossaries, code lists, and taxonomies are represented as customized XML files in many organisations. But this approach is attended with at least two problems, namely

- difficulties in exchange
- no direct tool applicability

The first problem occurs if one organisation wants to use another's vocabulary. Without a syntactical mapping, which has to be developed additionally, the vocabulary would not be usable by collaborating partners. The second problem is that existing components like reasoners cannot be used directly. The simpler case would be the inapplicability of standards-based search components, which need vocabularies for enhanced information-retrieval tasks. Both problems could be avoided by using a standard language for controlled vocabularies instead of developing difficult syntactical mappings.

In the next section, three standards accepted worldwide are described, covering all functionality of controlled vocabularies introduced in section 4.1.

#### 4.2.1 Resource Description Format

According to one of the early introductions [Las98] to semantics in the World Wide Web (WWW), the Resource Description Framework (RDF) [KC04] was not only influenced by the Web community itself, but also by various basic areas of computer science. Three areas in particular provided inspiration for the development of the framework

- Knowledge Representation
- Conceptual Modelling
- Document Modelling

The first area provided a mature and precise formalisation, but conceptual modelling also had to be taken into account in order to allow for schema-like constructions usable for controlled vocabularies of the type "D". Document-modelling issues played an important role due to the fact that electronic documents are central artefacts of the WWW that have to be equipped with semantic metadata.

Besides representation of classical metadata for electronic documents and representation of conceptual models, RDF is also able to represent arbitrary term or concept nets relevant for thesauri and glossaries. Whereas XML is used to model tree-based data, i.e. hierarchical structures (but not poly-hierarchies), RDF serves as some kind of 'lingua franca' for arbitrary information structures. In order to cope with such structures, RDF is a net-based language in contrast to XML.

RDF is based on three elements, namely

- Resources
- Literals
- Statements

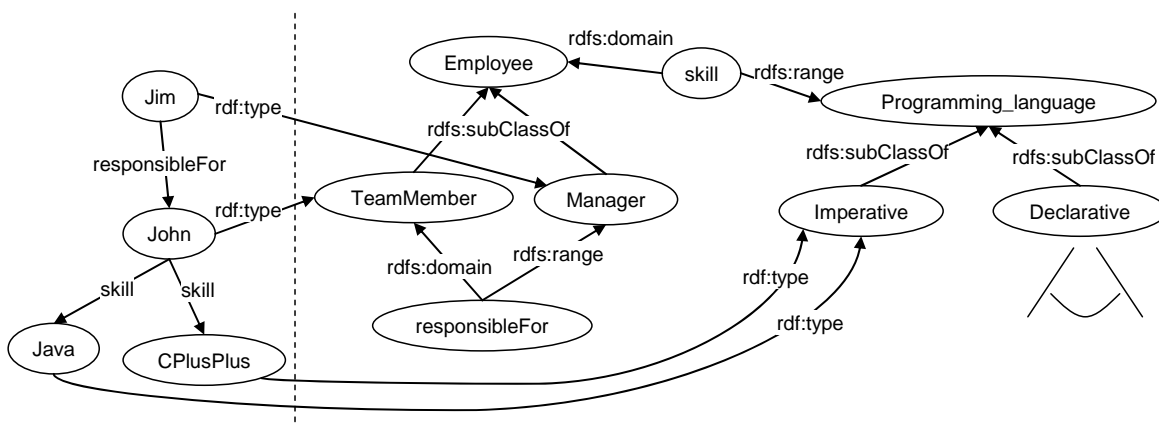


Figure 21: RDF and RDF Schema

Resources are the nodes of the information graph. Because RDF was intended to be a language that expresses semantic descriptions of Web resources, the main language construct is the URI reference (see [KC04]). These references are not only used to identify documents, but also to denote terms, relationships, and instances. In contrast to resources, which stand for ‘things’ of the world (whether material or immaterial), literals stand for themselves. They enclose numbers and strings and are comparable with simple types in a programming language.

RDF statements (also called RDF triples) are composed by means of resources and literals and have the form

- (*subject*, *property*, *object*)

where *subject* and *property* are a resource and *object* a resource or literal. A set of statements is called an RDF graph (or RDF model). On the left side of Figure 21, an RDF model is shown. Apart from references to the right part, the model consists of three statements, namely

- (*Jim*, *responsibleFor*, *John*)
- (*John*, *skill*, *Java*)
- (*John*, *skill*, *CPlusPlus*)

```
<owl:Class rdf:ID="MaleUnderage">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Class rdf:about="#Male"/>
        <owl:Class>
          <owl:complementOf rdf:resource="#Adult"/>
        </owl:Class>
      </owl:intersectionOf>
    </owl:Class>
  </owl:equivalentClass>
</owl:Class>
<owl:Class rdf:ID="MaleBaby">
  <rdfs:subClassOf rdf:resource="#MaleUnderage"/>
</owl:Class>
```

Figure 22: OWL fragment in RDF

On the right side, an RDF Schema is shown; again, it is represented by an RDF Model. Central constructs for defining schemata are the standard properties *rdfs:domain* and *rdfs:range*. Analogous to attributes in object-oriented programming, they define the source class (domain) and the target class (range). For example, *responsibleFor* is an attribute of managers targeting at team members. The *rdf:type* construct states the instance relationship as described for ontologies on page 40. As mentioned in conjunction with ontologies in section 4.1.4, one of the advantages of RDF is its flexibility to represent arbitrary term nets and information models.

#### 4.2.2 OWL

The Ontology Web Language OWL [BHH+04] was designed as a language to express full-fledged ontologies as described in section 4.1.5. OWL is defined on top of RDF and follows the triple representation. Therefore, every OWL ontology is again represented by a graph. Because the underlying formalism and semantics of OWL are given by Description Logics, the graph representation has to be able to handle special formulas like concept expressions and terminological

axioms. For example, the expressions from Table 3 have to be defined with RDF statements, as shown in Figure 22.

Description Logics not only had a strong influence on the development of the OWL standard. Also the frame paradigm, on which knowledge representation according to [GEF+99] is based, stands behind the OWL approach. The basic idea, which is also a key idea for object-oriented programming, is the attachment of properties to classes. Whereas in RDF properties can only be declared globally, OWL allows for declaring properties locally within the scope of a class definition.

This kind of localisation within OWL is called restriction. By means of restriction, ranges of global properties can be modified depending on the actual domain. The following example will exemplify this concept. Let the definition of the property *responsibleFor* be given by the RDF model in Figure 21. Assume additional classes *Parent* and *Child* and the request to define a property for parent responsibility.

In order to avoid duplicating and renaming the property *responsibleFor*, one can restrict it by the following fragment written in the Sparql/Turtle notation for expressing nested groups of statements [PS08].

```

▪ default:Parent
  rdfs:subClassOf
    [ rdf:type owl:Restriction ;
      owl:allValuesFrom default:Child ;
      owl:onProperty default:responsibleFor ] .

```

The above expression restricts the range of the property *responsibleFor* to children only if this property is used in conjunction with *Parent*.

The OWL standard defines several variants. The most flexible variant is OWL Full. OWL Full allows use of the complete RDF construct set. In contrast, OWL DL is more restricted and disallows the use of RDF constructs that destroy computational properties of Description Logics reasoners. The main goal of OWL DL is clarified in the following quotation: “*OWL DL was designed to support the existing Description Logics business segment and to provide a language subset that has desirable computational properties for reasoning systems*” ([BHH+04], section 1.2).

The main restriction of OWL DL is that classes and properties are not handled as ‘first-class citizens’. In other words, stating relationships between classes is not allowed. This restriction enables an efficient and complete reasoning according to the Description Logics semantics. Another variant is OWL Lite, which is even more restricted than OWL DL in that it generally disallows union and complement operators (see Table 3).

For some applications, it is required that classes can be treated as ‘first-class citizens’. According to [SM01], “... one may ask for questions like ‘show me the concept taxonomy including only those concepts for which you have some news in the last week’...” The reason why OWL DL is not adequate is simple: it would be necessary to define a property *classifiedWith* that ranges over concepts (nodes) from the taxonomy. A workaround for this problem could be that all taxonomic elements be defined as instances together with a project-defined subclass relation. However, this type of introduced relation would not benefit from all methods and algorithms already developed for the standard subclass relation. In other words, the central element of ontology languages would not be used.

The restriction described above is also unsuitable for meta-modelling tasks, according to [BHH+04], section 1.2: “*The complete OWL language (called OWL Full to distinguish it from the subsets) relaxes some of the constraints on OWL DL so as to make available features which may be of use to many database and knowledge representation systems, but which violate the constraints of Description Logic reasoners*”. Essentially, “*OWL Full will typically be useful for people who want to combine the expressivity of OWL with the flexibility and meta-modelling features of RDF*” ([BHH+04], section 8.1).



### 4.2.3 Thesauri Standard and SKOS

As described in section 4.1.3, thesauri mainly extend taxonomies by associative relationships and explicitly provide support for synonyms and homonyms. Although the standard [ANS05] is not a concrete language specification, it can be directly used as a language for representing thesauri. A mapping to a concrete syntax would be a one-to-one mapping. According to [ANS05], Table 5 shows the general concept introduced in section 4.1.3 and the related concrete thesaurus construct. Note that the thesaurus standard offers inverse relationships explicitly. In contrast to RDF and OWL, the set of relationships is fixed. Whereas RDF and OWL allow for freely definable relationships, the thesaurus standard subsumes such kinds of relationships under the associative relationship *related term*.

Relationship	Construct
Synonymy	<i>USE / UF (used for)</i>
Generic	<i>BTG (broader term generic) / NTG (narrower term generic)</i>
Instance	<i>BTI (broader term instance) / NTI (narrower term instance)</i>
Whole / Part	<i>BTP (broader term partitive) / NTP (narrower term partitive)</i>
Association	<i>RT (related term)</i>

*Table 5: Concrete thesauri constructs*

A concrete language specification that can be used to implement multilingual thesauri is the W3C standard SKOS (Simple Knowledge Organisation System [MB09]): “*SKOS - Simple Knowledge Organisation System - provides a model for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, folksonomies, and other similar types of controlled vocabulary*”. Its advantage is that it is completely based on RDF. This not only guarantees syntactical interoperability: if collaborating partners agree to use SKOS, it also allows for combining RDF schema constructs and OWL descriptions with thesauri constructs.

By combining SKOS elements with RDF and OWL elements, it is possible to customize the level of formalisation from the users’ perspective. As described in section 4.1.2, strong set semantics of *broader/narrower term* is sometimes not adequate. But if a concept in SKOS, say *animal*, is to follow the strong set semantics, it has to be defined by the statement

- *animal rdf:type rdfs:Class*

This statement allows for defining properties for the concept *animal*, e.g. *numberOfFeet*. Note that OWL DL is not able to express such structures due to the fact that classes are not ‘first-class citizens’.

Furthermore, two of the big advantages of SKOS are its intrinsic support for extensibility and basic multilinguality. In a dynamic environment like the Web, it is required that standard languages be adaptable to different needs that can change over the time. The extensibility of RDF as an underlying language is fully usable for the SKOS language. For instance, the default vocabulary of SKOS offers the RDF properties

- *skos:broader*
- *skos:narrower*

which can be extended or specialized to BTG or NTG if the application background is based on legacy thesauri using the construct set from Table 5.

Essentially, the big advantage of SKOS is the fact that it can be used as a representation language for all the kinds of controlled vocabularies introduced so far. Furthermore, it enables the combination of various RDF vocabularies in order to profit from language features available in every language branch.

#### MULTILINGUALITY

The concept of preferred terms that has been adopted is used to determine the representative of the synonymy equivalence class that is given with the construct

- *skos:prefLabel*

For representing synonyms, SKOS provides the construct

- *skos:altLabel*





Both the preferred and the alternative labels belong to the group of lexical labels. To support multilinguality, SKOS makes use of *language tags* introduced by RDF. In order to represent various language versions, lexical labels are furnished with these language tags. For instance, the fragment

- *animals*  
*rdf:type skos:Concept;*  
*skos:prefLabel "animals"@en;*  
*skos:altLabel "creatures"@en;*  
*skos:prefLabel "animaux"@fr;*  
*skos:altLabel "créatures"@fr.*

defines the concept *animal* with a preferred term and a synonym for English and French. In the context of SEMIC.EU, these representations of multilingual concepts are needed for resolving language-based semantic conflicts (see section 2.1.5).

#### 4.2.4 Relationships to Controlled Vocabulary Types

The following table shows the relationship between the type of controlled vocabulary and the standard language. Every language covers elements introduced in section 4.1 to the extent depicted by the arrow.

	Glossaries & Code Lists	Taxonomies	Thesauri	Ontologies	Full-fledged Ontologies
RDF					
OWL(-Full)					
Thesauri standard					
SKOS					

It has to be remarked that SKOS only defines a RDF-based meta-vocabulary for vocabularies of thesaurus type. But due to the fact that SKOS is extendible with customized meta-vocabularies because of its RDF foundation, it can be combined with other RDF-based languages like OWL such

that full-fledged ontologies can be represented. Note that SKOS is so far a W3C Candidate Recommendation but is expected to be a standard in future.

4.3 Development of Controlled Vocabularies

Analogous to software engineering and information modelling, methods for developing controlled vocabularies should be applied in order to guarantee their quality. Due to the fact that an ontology language, which is basically an amalgamation of the areas of conceptual modelling and knowledge representation, can be seen as a lingua franca for controlled vocabularies, principles and techniques for developing ontologies play an important role for the development of controlled vocabularies.

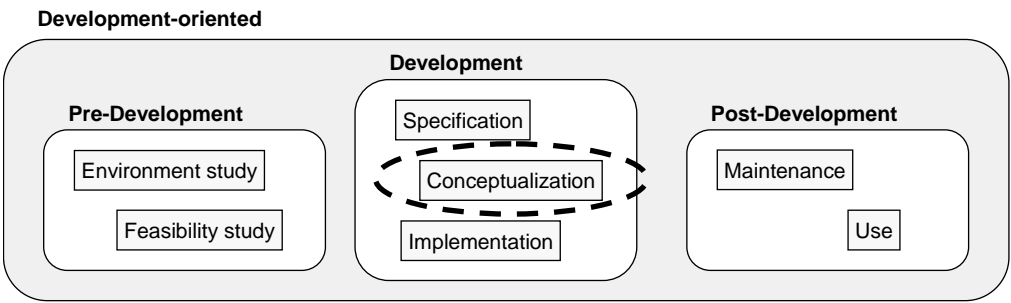


Figure 23: Activities of the Ontology Development Process

In order to exemplify the connection to the software engineering discipline, Figure 23 shows an adoption of the software engineering process [IEEE96] for the development of ontologies.

In view of the fact that semantic interoperability is one of the main goals of SEMIC.EU, especially in regard to the platform’s ability to handle semantic conflicts, conceptualisation can be seen as the core activity (as depicted by the dashed line in Figure 23). This is because the conceptualisation activity gradually encompasses all language elements of controlled vocabularies used for handling the respective conflict types.

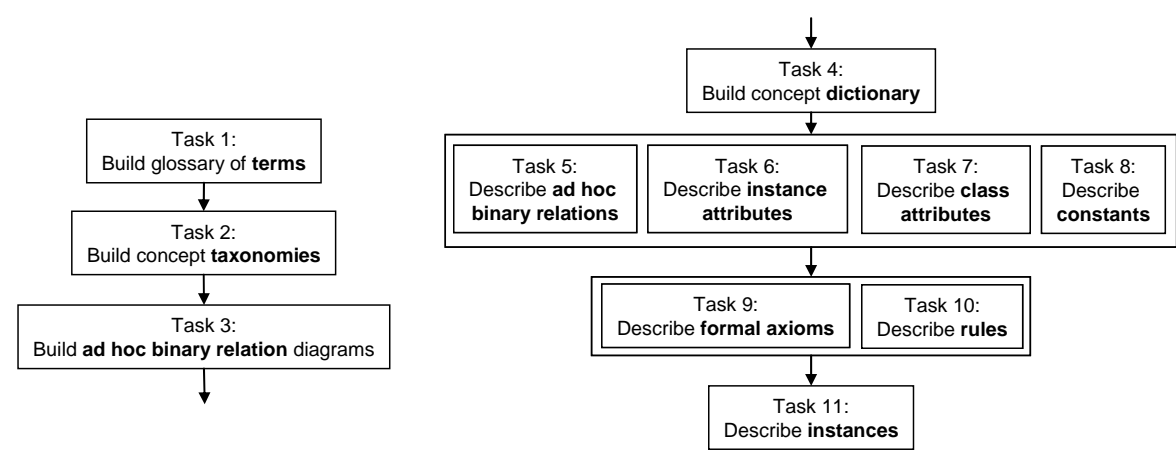


Figure 24: Conceptualisation tasks according to [FGJ97]

The following section describes the development process in more detail by means of the Methontology approach [FGJ97], its being one of the most prominent methods in this type of activity. In order to reflect the relationship between interoperability and the tasks of the conceptualisation activity, a table that summarizes the correspondence between conflict types and tasks is presented at the end of this section.

#### CONCEPTUALISATION AS THE CORE ACTIVITY

The conceptualisation activity consists of eleven tasks, starting from simple structures and ending with elements of full-fledged ontologies and instance sets (see Figure 24). The conceptualisation activity starts with building a glossary of terms, which corresponds to the first vocabulary type described in section 4.1.1.

The glossary is usually organized in tabular form. Besides the term itself, a list of synonyms and acronyms can be defined, and a very short natural-language description has to be provided. In the case of multilinguality, language versions will be included within this task (see SKOS multilinguality in section 4.2.3). At this stage, relationship terms are on the same level and not attached to concepts. Additionally, the classification of the term can be stated in order to be prepared for transition to ontologies that make a distinction between concepts, relationships of simple types (attributes), and relationships between concepts. This is the first stage necessary for handling synonym conflicts. Homonym conflicts can also be handled by means of this stage if term qualifications are provided.

The second task is to build taxonomies according to the factors described in section 4.1.2. The more precise the taxonomy has to be, the more care required in considering the set semantics. This means that the issue of whether the proper subclass relationship can be applied has to be checked by asking if everything denoted by  $X$  is also denoted by  $Y$ , where  $X$  is the specialisation and  $Y$  the more general concept.

For instance, it is clear that every team member (every individual denoted by the concept ‘team member’) is also an employee. Not all persons, however, are employees. With respect to semantic conflicts, from this point on, sufficient information has been described to make it possible to handle hypernym and generalisation conflicts. The more precise the subclass relation is defined in the sense of set semantics, the more accurate the hypernym conflict detection can be performed.

The third task describes a rough overview over non-taxonomic relationships between concepts in diagrammatic form, where concepts are depicted as nodes and relationship as arcs between them. This task corresponds to vocabulary type *thesauri* and partly to the type *ontologies* because relationships are named as in ontologies and not only indicated as related terms as in thesauri.

Task 4 sums up the tasks performed so far in tabular form, yielding a so-called “concept dictionary”. For each concept, relationships are also assigned where the following items are stated:

- name
- attributes
- relationships (with range)
- instances (optional)

Task 5-8 takes care of more detailed descriptions of the conceptualisation produced so far. In task 5, relationships are refined based on diagrams stated in task 3. At least the domain and range of a relationship, i.e. the source and target concept, are specified. Additionally, if classical data modelling should also be covered by the conceptual model, attributes and constants are specified in detail in task 6-8. A reference to the concept it belongs to, as well as value types with possible measurement units and precisions should be stated. At this, attributes’ values valid for the whole concept (class attributes) are also considered.

Due to the fact that schema elements, namely attributes and relationships with domain and range, are defined in task 4 and refined in task 5-8, the results of task 4-8 yields an ontology. It is exactly this additional information compared to thesauri that makes possible the handling of higher-level conflicts like schema isomorphism conflicts (see section 4.1.4).

Task 9 and 10 specify axioms and rules as described in section 4.1.5 by means of a consistent textual notation, usually in form of constraint formulas or if-then rules, respectively. In addition to the rule itself, a name and a natural-language description have to be stated. Furthermore, concepts, attributes, and relationships referred to have to be quoted, and variables used have to be specified. In the last (optional) task, instances are listed together with their attribute values.

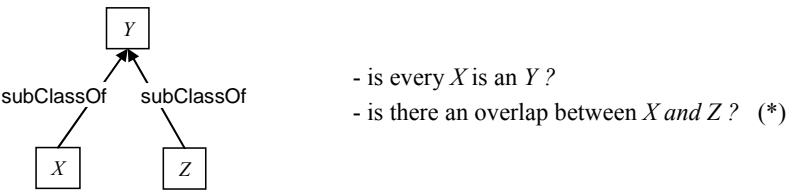


Figure 25: Questions for subclass relationship

The purpose of task 9 and 10 with respect to semantic conflict handling is to state relationships between concepts based on a set-theoretic view that can be used in order to detect overlapping conflicts (see section 2.2.1). Figure 25 shows a question (\*) related to the subclass construct and relevant for overlapping conflicts. Whereas the first question has to be asked in general for taxonomies (task 2), the second question is important for defining axioms. For instance, ‘no’ as an answer for the second question yields the axiom *X and Z equals empty* (see Table 3), which can be used for automatically detecting overlapping conflicts.

CONFLICT-RELATED TASKS

conflict type		task				
		1	2	3	4 - 8	9 - 11
data level	synonyms	↔				
	homonyms	↔				
	data language	↔				
schema level	synonyms	↔				
	homonyms	↔				
	hypernyms		↔			
	overlapping					↔
	schema isomorphism			⏏↔		
	generalisation		↔			

The table above summarizes the correspondence between conflict types and tasks. The first task develops elements of the controlled vocabulary that can be used by matching and mapping techniques to handle the first five conflict types. Taxonomic elements supporting the handling of hypernym and

generalisation conflicts are created in task 2. Task 3-8 develops elements, namely named associative relationships, in task 3 and schema elements like properties with range and domain in task 4-8, which can be used in conjunction with schema isomorphism conflicts. Finally, overlapping issues are supported by axioms and thereby covered in task 9-11.

The methodologies described in the last two sections aim at detecting and resolving semantic conflicts. However, the preferred way to handle semantic conflicts is to avoid them in the first line or at least to reuse existing solutions for resolving them.

The structuring of assets has an impact on the occurrence of semantic conflicts. In order to achieve an appropriate structuring one should apply suited principles of modularisation, which are presented in the following section. In particular, it is described how these principles, besides supporting SEMIC.EU's general purpose of reusing Semantic Interoperability Assets, support avoiding semantic conflicts, how they ease detecting semantic conflicts, and how they support the reuse of existing solutions for resolving semantic conflicts.

## 5. MODULARISATION

The following section discusses principles of modularisation, focussing on the essential goals and purposes of SEMIC.EU, i.e. achieving reusability of Semantic Interoperability Assets and achieving semantic interoperability by avoiding or detecting and resolving semantic conflicts.

Modularisation is a common concept in several engineering disciplines, in particular in the field of software engineering. This section outlines how well-known software engineering principles of modularisation should be applied to the development of the artefacts of a Semantic Interoperability Asset, i.e. in particular the schemata and controlled vocabularies within an asset. Applying these principles supports avoiding or detecting and resolving semantic conflicts mainly for two reasons:

- Modularisation essentially reduces potential conflicts by reusing proven existing assets and thus restricts the occurrence of conflicts to be resolved to the new, not reused, parts of a schema or a controlled vocabulary.
- Modularisation reduces potential conflicts by separating unspecific from specific aspects of a schema or controlled vocabulary and thus reduces or “localises” conflicts.

Finally, the section outlines the relation between reusability and granularity of an asset with respect to “rightsizing” and parameterisation.

### 5.1 Modularisation Principles

The general principle of modularisation is separation of concerns, which can be applied to quite different concerns. The following subsections will discuss different applications of this general principle.

#### 5.1.1 Structured Data Types

On the atomic level of data modelling, different properties of an entity are modelled as different attributes of it. What is considered as atomic, however, depends on the context. Applied to the structured data type related to a person, an academic title can be modelled as a separate attribute or be considered as part of the name, though – at least in some countries – academic titles are actually considered to be part of the name.

Moreover, an address can be modelled as one attribute. In this case, the value of the attribute may be the whole address of a person, including the name. When this attribute is only used for delivering mail to the related person, this is an appropriate modelling of an address.

When, however, the address is to be related to other data for a statistical evaluation, e.g. to determine the average income in a city, it would be necessary to extract the corresponding information from the address.

Otherwise, for different structures of addresses in different countries, an exchange of data including addresses between these countries would require a schema mapping between the structures to resolve semantic conflicts, whereas data exchange is much easier for addresses modelled as one attribute.

#### REUSABILITY

From the SEMIC.EU point of view, reusability of a structured data type in a pan-European context means that its structure fulfils the requirements of each of the Member States. This is best achieved by the highest extent of granularity possible for those parts of the data type that are common to each of the Member States, e.g. the postal code in an address. These parts can and should be modelled as

mandatory, as each address has to contain a postal code. Other attributes, which do not occur in other Member States, should be modelled as optional.

The appropriate extent of granularity depends on the possible contexts in which the data is to be used. A data type that has to be reusable in many different contexts and domains, e.g. a data type like address, should be structured as granularly as possible, as it is supposed that there are (or at least will be) contexts that require the highest degree of granularity possible.

Other more specific data types, e.g. data types restricted to a specific domain, may be structured less granularly if the internal structure of the corresponding data values does not matter in the pan-European context.

The appropriate extent of granularity for a specific context is mainly determined by non-technical requirements and constraints, e.g. legal requirements that are out of the scope of this study.

#### AVOIDING SEMANTIC CONFLICTS

As a general rule in regard to the avoidance of semantic conflicts, the more different structures of a data type that capture the same information are involved, the more conflicts occur. As soon as different structures, implemented in different schemata, are involved, mappings between these schemata have to be provided. In general, resolving semantic conflicts implied by a mapping from a fine-grained to a coarse-grained schema is possible. Vice versa, semantic conflicts implied by 1:N, M:N or higher-order correspondences occur (see section 3.1.1). These conflicts can only be resolved when appropriate extraction functions are available that resolve these correspondences (see Figure 3 and 4). Such semantic conflicts cannot be resolved in each case, e.g. it is not possible to extract first name and last name from a name value when they may appear in an arbitrary order within a name. Thus, a mapping between different schemata is only possible if the schemata involved have the same granularity, either directly or after applying appropriate extraction functions.

In a pan-European context where schemata of each of the Member States are involved, different structures of schemata and related mappings imply that the data can only be exchanged on the basis of an integrated schema. However, it is also not possible to resolve semantic conflicts implied by a different structure of the schemata involved that cannot be resolved by means of a point-to-point mapping by a corresponding two-step mapping via an integrated schema (see section 3.2). Such semantic conflicts have to be accepted, or they have to be “resolved” on a political level, i.e. agreeing on a common standard to which the Member States have to adhere.

#### 5.1.2 Aggregation/Composition

Aggregation and composition correspond to structured data types on a higher level. Both aggregation and composition apply when entities are combined with larger units. The difference between them is that in a composition, the existence of a part depends on the existence of the composed unit. In an aggregation, the parts exist independently from the aggregation, e.g. a consortium consists of its members, but each of the members will continue to exist after the consortium has been dissolved.

Composition and aggregation are means to model logically independent parts of information in separate structures. For example, the information related to a person’s address and bank account should be modelled in separate (sub-)schemata.

#### REUSABILITY

The different sub-schemata of an aggregated schema can be reused independently. The reusability of each sub-schema is higher than the reusability of a schema that captures the same information, but is implemented as a monolith. For example, if only an address is needed, a schema comprising address and bank account would have to be modified, whereas a sub-schema address could be reused without modification.



A further advantage of modelling logically separate parts in separate structures is that modifications of one schema usually do not imply modifications of another. For example, a modification from a four-digit postal code to a five-digit one does not imply any modification to the person's bank account.

In case of a pan-European data exchange, for a modular integrated schema within the mappings from and to each of the national data formats, a modification from four-digit to five-digit postal codes would imply a modification of the address module including the related mappings, but not a modification of other modules and mappings.

#### AVOIDING SEMANTIC CONFLICTS

As modular schemata have a higher potential for reusing sub-schemata than monolithic ones, they support avoiding semantic conflicts. For example, two heterogeneous modular schemata for the exchange of data related to an employee may reuse a sub-schema for addresses. Thus, for a data exchange based on these heterogeneous schemata, no semantic conflicts occur concerning address data. Thus, the number of semantic conflicts is lower than compared to a data exchange based on two completely heterogeneous schemata.

A similar result can be achieved by reusing a common integrated schema, including mappings to the local schemata. For example, a German modular schema uses a German address subschema, for which an integrated address schema including a mapping from the German schema to a French address schema exists. For a data exchange between Germany and France that is performed via the integrated schema, the semantic conflicts related to the exchange of address data are resolved within the integrated address schema and its mappings. Thus, existing solutions for resolving semantic conflicts are reused.

Moreover, even for heterogeneous modularised schemata, detecting and resolving semantic conflicts is easier, as the semantic conflicts are more restricted to the corresponding modules of the modular schemata involved than for two monolithic ones.

#### 5.1.3 Generalisation/Specialisation

Generalisation/specialisation is a concept to model common attributes of different information entities in a separate structure. For example, a bike can be a mountain bike, a road bike, or a tandem bike. Each of these types of bike can be modelled as different concepts in different structures. Using generalisation, all the common attributes can be modelled in one structure, whereas all attributes specific to a type can be modelled in a separate structure for each of them.

#### REUSABILITY

Generalisation/specialisation fosters reuse, as a schema comprising the common parts of variants of a concept is more reusable than the more specific schemata for each of the variants. For example, a schema for a bike can be used for modelling a new type of bike by merely adding the specific attributes in a separate sub-schema, whereas a schema for a mountain bike is not reusable without modification to model a new type of bike.

Moreover, modifications of a schema of a subtype do not imply modifications of another schema. Adding new sub-types and their schemata, respectively, can be done without affecting the schemata of the other subtypes and those of the general type. Due to the general character of the information captured, the sub-schema comprising the common characteristics has the highest potential for being reused.

#### AVOIDING SEMANTIC CONFLICTS

Similar to aggregation, the sub-schemata of a modular schema constructed concerning generalisation/specialisation have a much higher potential for being reused compared to a monolithic schema. Therefore, they support avoiding semantic conflicts.

Reusing a common integrated schema implies reusing existing solutions for resolving semantic conflicts. The same logic as depicted in the subsection on aggregation applies.

##### 5.1.4 Application Dependency

The example introduced in subsection 5.1.3 illustrates the application dependency of modularisation. Modelling an address as one attribute is appropriate when it is only used for delivering mail, whereas a structured schema is appropriate if access to the parts of an address is needed.

Generally, separation of concerns in regard to a single application should mainly separate aspects and parts specific to the application from those aspects common within the domain and other applications within the domain, respectively.

Even application-specific schemata may be separated. For example, an application that has to exchange data with others, implying different messages to these applications, may separate parts that are common in all these messages from parts that differ.

#### REUSABILITY

Analogous to aggregation and generalisation, sub-schemata can be easier reused without modifications than monolithic schemata. The more general application-independent sub-schemata usually have a higher potential for being reused in comparison to the sub-schemata comprising the application-specific characteristics.

#### AVOIDING SEMANTIC CONFLICTS

Reuse of application-independent sub-schemata within the application supports the avoidance of semantic conflicts, the reuse of existing solutions for resolving semantic conflicts, and the detection of semantic conflicts (see subsection 5.1.2).

##### 5.1.5 Domain Dependency

Different domains usually use different terminologies and concepts, e.g. even addresses may vary in different domains. Otherwise, particularly in a cross-domain context, commonalities in concepts and terminologies should be identified and separated from those parts of data exchange formats as well as of controlled vocabularies that are domain-specific.

As a general rule in a cross-domain scenario, modifications necessary in one domain are independent from the other. Thus, these modifications do not imply modifications of schemata related to the other domain.

#### REUSABILITY

In general, sub-schemata can be easier reused without modifications than monolithic schemata. The domain-specific sub-schemata have a potential for being reused within their domains, whereas monolithic schemata mixing aspects of different domains may be hardly reusable.

#### AVOIDING SEMANTIC CONFLICTS

Reuse of domain-specific sub-schemata supports the avoidance of semantic conflicts, the reuse of existing solutions for resolving semantic conflicts, and the detection of semantic conflicts (see subsection 5.1.2).

##### 5.1.6 Domain Independency

There are certain infrastructural aspects of data exchange that are independent of the domain in which they occur. For example, federated identity and access-management are crucial issues in the eHealth as well as the eJustice domain: the security requirements of data exchange are high in both. Nevertheless, the technical means of secure data exchange, e.g. application and profiling of relevant standards as SAML, is at least partly independent from the application domain.

Data-exchange formats and related specifications, including components that are at least partly domain independent, should be modelled or profiled in such a way that the domain-independent parts are separated to the greatest extent possible. For example, in an identity and access-management scenario, a generic role concept should be developed. This concept should abstract from domain-specific roles (such as physician or patient in the eHealth domain), but relate the roles to different levels of security and access rights. Such a concept could be adapted or parameterised to suit to other domains with similar (high) requirements for secure data exchange.

Other, at least potentially domain-independent information units to be modelled in separate schemata are low-level units (such as addresses, etc.), for which standardisation efforts have been established in many of the Member States. Standardisation efforts have also been established for infrastructure frameworks (such as the identity and access-management framework mentioned above).

#### REUSABILITY

Domain-independent schemata (low-level ones as well as complete frameworks) have the highest potential for reuse in the SEMIC.EU context.

#### AVOIDING SEMANTIC CONFLICTS

Domain-independent sub-schemata should as far as possible be reused in order to support the avoidance of semantic conflicts, the reuse of existing solutions for resolving semantic conflicts, and the detection of semantic conflicts (see subsection 5.1.2).

##### 5.1.7 Invariance

Another criterion guiding the decision on how to modularise a schema is which of its parts are subject to frequent changes and which stay constant. When such a differentiation can be identified, the corresponding parts should be separated. Moreover, invariant and more volatile parts in different schemata can be implemented differently. For example, an invariant, fixed set of values can be implemented via a code list.

#### REUSABILITY

Analogous to the principles described above, sub-schemata can be easier reused without modifications than monolithic schemata. In particular, sub-schemas that cover invariant characteristics have a higher potential for being reused.

#### AVOIDING SEMANTIC CONFLICTS

In addition to the general advantages of modularisation for avoiding semantic conflicts, as described above (see subsection 5.1.2), using code lists avoids data-value conflicts (see section 2.1.1), wrong values, and typing errors.

## 5.2 Reuse and Granularity

Concerning the relations between reuse and granularity, several factors that have particular impact on reusability and the benefit of reuse can be identified.

In general, reusability is positively correlated to generality. The more general a schema, the more reusable it is. Generality is inversely correlated to complexity. On the one hand, the more complex a schema, the more specific information it includes and, therefore, the less general it is. On the other hand, the more complex and larger a schema is, the greater the benefit of reusing it compared to reusing numerous small schemata, unless significant modifications and adjustments of the schema are necessary.

The following two approaches should be applied in order to increase

- the benefit of reusing small schemata
- the generality of large schemata

### 5.2.1 Bundling of Small Schemata

Modularisation of a large schema by applying the principles described above results in modular schemata composed of sub-schemata. The same result can be achieved by combining small schemata, thus producing a composed modular schema. For example, an exchange of data related to an employee includes address and bank account data. A data exchange related to the customer of an online shop includes address data and may include bank account data, as well. A composed schema, including a sub-schema for address data and a sub-schema for bank account data, could be reused in both cases.

The larger such schemata composed of basic building blocks are, the larger the benefit of reusing them, as opposed to reusing a lot of small, unrelated schemata.

### 5.2.2 Parameterisation of Large Schemata

As introduced in subsection 5.1.6, the generality of a schema or a system of modular schemata can be improved by parameterisation concepts. For an identity and access-management scenario, a generic role concept should be used that abstracts from domain-specific roles. Instead, different levels of security and access rights should be defined that fulfil different levels of security requirements, e.g. a high level of security in the domain of eJustice and eHealth, and lower levels for other domains.

## 6. CONCLUSIONS

This study introduced methodologies facilitating achieving semantic interoperability and increasing the potential to reuse Semantic Interoperability Assets. Insufficient semantic interoperability is characterised by the presence of semantic conflicts. These semantic conflicts occur, when the sender and receiver use different data structures when exchanging data. The following steps are considered to be appropriate for achieving semantic interoperability:

- Semantic conflicts should be avoided as far as possible, e.g. by restricting the possible values in an XML document to a fixed set defined in a code list to avoid ambiguity.
- When semantic conflicts arise, they should be resolved when possible, e.g. by translating data values from one language to another.
- In situations where semantic conflicts cannot be resolved, e.g. when a message contains numerical values of lower precision than needed by the receiver, as far as such a conflict can be detected at all, it should be noted and documented appropriately.

On a pan-European level, one has to deal with the existence of semantic conflicts. Bearing in mind the high degree of autonomy of the individual Member States, it may be very hard and time-consuming to agree on a common standard. Besides, the use of common standards cannot be enforced on a pan-European level at all. Hence, data exchange on a pan-European level currently in most cases is based on different source/target schemata. This necessarily implies the presence of semantic conflicts.

This study presented in some detail appropriate state-of-the-art methodologies for treating these conflicts, i.e. detect and resolve the conflicts as far as possible. As a general approach, the local solutions used in the respective Member States are harmonised with each other in order to provide for semantic interoperability on a pan-European level. From a technical point of view, this means dealing with heterogeneous schemata on a practical basis.

These state-of-the-art methodologies consist of:

- classification of semantic conflicts,
- schema mapping,
- schema integration,
- schema matching,
- controlled vocabularies, and
- modularisation principles.

As a foundation for applying the concrete methodologies to resolve semantic conflicts, a classification of these conflicts is introduced. Semantic conflicts may exist on the data level as well as on the structure of the schemata used when exchanging data. For each kind of semantic conflict, specific techniques to resolve this conflict exist and may be applied.

Schema mapping provides these specific techniques to address each of the semantic conflicts described in the study. The core of schema mapping is the identification of those sub-structures in the source and the target schema that capture the same, i.e. semantically equivalent information. These sub-structures are related by correspondences. Usually, these correspondences define more than one possible mapping suited to resolve the semantic conflicts caused by the heterogeneity of source/target schemata. A domain expert has to decide which of the possible alternative correspondences should be chosen. Based on this decision, a corresponding data transformation is derived.

In general, schema mapping in practice should be performed with the aid of a schema mapping tool, in particular for generating the data transformation.

For mappings between more than three schemata – the usual case in the pan-European context –, the study presents the methodology of schema integration. Applying this methodology offers a way to contain the otherwise exponentially increasing number of mappings necessary when relying on direct point-to-point mappings between each source and each target schema.

The techniques presented in the study compile a number of different schemata into one integrated schema built to logically subsume all the source schemata. In contrast to (simple) schema mapping, a mapping from a source schema to a target schema is split up into two separate mappings, the first mapping linking the source to the integrated schema, the second mapping linking the integrated schema to the target schema. Accordingly, two related data transformations are derived. The mapping techniques used in the integration case are the same as the ones used in the simple case.

Both schema mapping and schema integration rely on detecting semantic correspondences. The schema matching techniques introduced in this study feature the automatic detection of these semantic correspondences. Such an automatic detection is indispensable if large and complex schemata have to be mapped or integrated.

Matching techniques exist on the schema as well as on the data level. These techniques use different methodologies for comparing names and different structural means, e.g. comparing of granularity, cardinality and constraints.

Using controlled vocabularies substantially supports the mapping and matching of schemata by making it possible to detect correspondences and semantic conflicts. This study describes controlled vocabularies in conjunction with a classification using their respective expressiveness as a criterion. One of the primary results is that controlled vocabularies are a necessary means for handling semantic conflicts due to their suitability to clarify the semantics of terms used in schemata. It has been shown that conflict handling may be greatly supported and improved by choosing more expressive vocabulary types.

In order to use a specific type of controlled vocabulary in a concrete project, it is necessary to represent the vocabulary chosen in a machine understandable form. To this end, the study introduces concrete languages for controlled vocabularies and their relationship to vocabulary types.

The presentation of controlled vocabularies concludes with a description of the development process of ontologies as the most general form of controlled vocabularies. In particular, the relationship between the task within the process and its contribution to handling a specific semantic conflict type is established.

The table below summarises the findings of the study in respect to individual methodologies' ability to adequately handle the different types of semantic conflicts.

conflict type		support of						
		Schema Mapping	Schema Matching	Controlled Vocabularies				
				A	B	C	D	E
data level	synonyms	+	+	+	+	+	+	+
	homonyms	-	+	o	o	+	+	+
	data representation	+	+					
	data unit	+	+					
	data precision	o	o					
	data language	+	+	+	+	+	+	+

2009-06-02

schema level	synonyms	+	+	+	+	+	+	+
	homonyms	+	+	o	o	+	+	+
	hypernyms	o	+	-	+	+	+	+
	overlapping	o	o	-	-	-	-	+
	schema isomorphism	+	-	-	-	-	+	+
	missing entity	o	-					
	data type	o	+					
	generalisation	o	o	-	+	+	+	+
	schematic discrepancies	+	-					
	hierarchy	+	-					
	Foreign key	+	-					
	constraints	o	+					
- : no support   o : conditional support   + : support A: Glossaries, B: Taxonomies, C: Thesauri, D: Ontologies, E: Full-fledged Ontologies								

The preferred option to handle semantic conflicts is to avoid them in the first line. This may be achieved by the use of:

- Standardised schemata – well-known examples of this approach are the efforts aimed at developing basic and potentially self-contained information units in the form of Core Components or XML schemata, e.g. address, bank account, etc. Those are used frequently in real-world applications in many of the Member States on the one hand and the international HL7 standard in the eHealth domain on the other.
- Controlled vocabularies – for instance, code lists that might be integrated into XML schemata to restrict the set of possible values, can be used to avoid ambiguity, wrong values, cryptic abbreviations, and typing errors.
- Appropriate modularisation principles that help to avoid conflicts because it boosts the reuse of existing or even formerly standardised schemata.

The modularisation principles introduced and discussed in this study serve two fundamental purposes: firstly, they increase reusability of Semantic Interoperability Assets. Secondly, they increase semantic interoperability by avoiding semantic conflicts, mainly by reusing existing modular assets and existing resolutions of semantic conflicts. These modularisation principles are primarily based on the separation of different asset and artefact concerns or dimensions, such as separating specific from unspecific aspects, domain-specific from domain-independent aspects, invariant from variant aspects, etc.

Furthermore, the approaches of bundling and parameterising of modular schemata introduced in this study provide additional means to facilitate the development of reusable Semantic Interoperability Assets.

In accordance with the preferred approach to avoid semantic conflicts, standardisation efforts have been launched in most of the Member States. One of the most widely-used methodologies on a practical level is represented by the Core Components Technical Specification (CCTS) [UNC03]. Originally, CCTS is aimed at information interoperability in the e-business sector. It provides a methodology for developing self-contained semantic building blocks that represent the general types

of business data. Subsequently, the focus of the CCTS is clearly on providing suitable means for a standardisation of those building blocks. The primary goal of the concept of Core Components is the particular reuse of standardised data units for common types of business data.

So far, standardisation efforts are mainly restricted to the national level. An effort to develop pan-European Core Components would be a significant contribution to avoid semantic conflicts in the pan-European context addressed by SEMIC.EU. The current situation of facilitating the evolution of rather national solutions towards a potentially pan-European operation, however, may greatly benefit by the application of the concrete methodologies proposed through this study.



**Appendix A REFERENCES AND LITERATURE**

- [ANS05] ANSI/NISO Z39.19 - Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies, National Information Standards Organisation, 2005
- [Atr90] Atran, S. Cognitive foundations of natural history: towards an anthropology of science. Cambridge University Press. 1990.
- [BCM+03] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F. The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press. 2003.
- [BHH+04] Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., and Stein, L. A. OWL Web Ontology Language Reference. W3C Recommendation. W3C Consortium. 2004.
- [BLN86] Batini, C., Lenzerin, M., and Navathe, S. B. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4). 1986.
- [BMPQ04] Bernstein, P. A., Melnik, S., Petropoulos, M., and Quix, C. Industrial-strength schema matching. *SIGMOD Rec.* 33, 4. 2004.
- [Boa07] Boag, S. et al. (eds). XQuery 1.0: An XML Query Language. W3C. 2007.
- [Cab09] Cabibbo, L. On Keys, Foreign Keys and Nullable Attributes in Relational Mapping Systems. In *Proceedings of the 12th international Conference on Extending Database Technology: Advances in Database Technology* (Saint Petersburg, Russia, March 24 - 26, 2009). M. Kersten, B. Novikov, J. Teubner, V. Polutin, and S. Manegold, Eds. EDBT '09, vol. 360. ACM, New York, NY. 2009.
- [Che76] Chen, P. P. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)* 1, 1. 1976.
- [Cod70] Codd, E. F. A relational model of data for large shared data banks. *Communications of the ACM* 13, 6. 1970.
- [CT06] Chiticariu, L. and Tan, W.-C. Debugging schema mappings with routes. In *Proceedings of the 32nd international Conference on Very Large Data Bases* (Seoul, Korea, September 12 - 15, 2006). U. Dayal, K. Whang, D. Lomet, G. Alonso, G. Lohman, M. Kersten, S. K. Cha, and Y. Kim, Eds. Very Large Data Bases. VLDB Endowment. 2006.
- [DBH07] Duchateau, F., Bellahsene, Z., and Hunt, E. XBenchMatch: a benchmark for XML schema matching tools. In *Proceedings of the 33rd international Conference on Very Large Data Bases* (Vienna, Austria, September 23 - 27, 2007). Very Large Data Bases. VLDB Endowment. 2007.
- [DR02] Do, H. and Rahm, E. COMA: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international Conference on Very Large Data Bases* (Hong Kong, China, August 20 - 23, 2002). Very Large Data Bases. VLDB Endowment. 2002.
- [FGJ97] Fernández, M., Gómez-Pérez, A., and Juristo, N. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. Symposium on Ontological Engineering of AAAI. Stanford (California). 1997.
- [FI08] Fraunhofer ISST and Jinit[. Study on multilingualism. European Commission – IDABC. 2008.
- [GEF+99] Grosso, W. E., Eriksson, H., Fergerson, R. W., Gennari, J. H., Tu, S. W., and Musen, M. A. Knowledge modeling at the millennium (the design and evolution of Protege-2000). In *Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW99)*. 1999.
- [Gloss] Glossarist.com: a searchable and categorised directory of glossaries and topical dictionaries. <http://www.glossarist.com/> (last visited 08.04.2009)

- 
- [Gru08] Gruber, T. R. *Ontology*. Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag. 2008.
- [Hel06] Held, B. *Infrastructure for pan-European Semantic Interoperability: The IDABC XML Clearinghouse*. 2006.
- [HHH+05] Haas, L. M., Hernández, M. A., Ho, H., Popa, L., and Roth, M. Clio grows up: from research prototype to industrial tool. In *Proceedings of the 2005 ACM SIGMOD international Conference on Management of Data* (Baltimore, Maryland, June 14 - 16, 2005). SIGMOD '05. ACM, New York, NY. 2005.
- [IEEE96] IEEE Standard for Developing Software Life Cycle Processes. IEEE Computer Society. New York (USA). 1996.
- [KC04] Klyne, G., and Carroll, J. J. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. W3C Consortium. 2004.
- [Las98] Lassila, O. 1998. Web Metadata: A Matter of Semantics. *IEEE Internet Computing* 2, 4. 1998.
- [Lev65] Levenshtein, V. I. Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission* 1(1). 1965.
- [LN06] Legler, F. and Naumann, F. A. *Classification of Schema Mappings and Analysis of Mapping Tools – Extended Version*. Technical report, Hasso-Plattner-Institut, Universität Potsdam. 2006.
- [LN07] Leser, U. and Naumann, F. *Information Integration* (in German). Heidelberg: dpunkt.verlag, 2007.
- [LS08] Libkin, L. and C. Sirangelo. Data exchange and schema mappings in open and closed worlds. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Vancouver, Canada, June 09 - 12, 2008). PODS '08. ACM, New York, NY. 2008.
- [MB09] Miles, A. and Bechhofer, S. *SKOS Simple Knowledge Organisation System Reference*. W3C Consortium. 2009.
- [MBR01] Madhavan, J., Bernstein, P. A., and Rahm, E. Generic Schema Matching with Cupid. In *Proceedings of the 27th international Conference on Very Large Data Bases* (Sept. 11 - 14, 2001). P. M. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, Eds. Very Large Data Bases. Morgan Kaufmann Publishers, San Francisco, CA. 2001.
- [MGR02] Melnik, S., Garcia-Molina, H., and Rahm, E. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proceedings of the 18th international Conference on Data Engineering* (February 26 - March 01, 2002). ICDE. IEEE Computer Society, Washington, DC. 2002.
- [OR23] Ogden, C. K. and Richards, I. A. *The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism*. London: Kegan Paul. 1923.
- [Pan08] Pankowski, T. XML Schema Mappings Using Schema Constraints and Skolem Functions. *Studies in Computational Intelligence (SCI)* 102. 2008.
- [Pry99] Prytherch, R. *Harrod's Librarians' Glossary and Reference Book*. Gower Publishing Company. 1999.
- [PR04] Park, J. and Ram, S. Information Systems Interoperability: What Lies Beneath? *ACM Transactions on Information Systems*, Vol. 22, No. 4. 2004.
- [PS08] Prud'hommeaux, E. and Seaborne, A. *SPARQL Query Language for RDF*. W3C Recommendation. W3C Consortium. 2008.

- [PVH+02] Popa, L., Velegrakis, Y., Hernández, M. A., Miller, R. J., and Fagin, R. Translating web data. In *Proceedings of the 28th international Conference on Very Large Data Bases* (Hong Kong, China, August 20 - 23, 2002). Very Large Data Bases. VLDB Endowment. 2002.
- [RB01] Rahm, E. and Bernstein, P. A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 4. 2001.
- [SK93] Sheth, A. P. and Kashyap, V. So Far (Schematically) yet So Near (Semantically). In *Proceedings of the IFIP WG 2.6 Database Semantics Conference on interoperable Database Systems (Ds-5)* (November 16 - 20, 1992). D. K. Hsiao, E. J. Neuhold, and R. Sacks-Davis, Eds. IFIP Transactions, vol. A-25. North-Holland Publishing Co., Amsterdam, The Netherlands. 1993.
- [SM01] Staab, S. and Maedche, A. Knowledge portals: Ontologies at work. *AI Magazine*, 22(2). 2001.
- [Sow99] Sowa, J. F. Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing. 1999.
- [SPD92] Spaccapietra, S., Parent, C., and Dupont, Y. Model independent assertions for integration of heterogeneous schemas. *The VLDB Journal* 1, 1. 1992.
- [TTPL07] Tambouris, E., Tarabanis, K., Peristeras, V., and Liotas, N. Study on Interoperability at Local and Regional Level. Final Version – Version 2.0. eGovernment Unit DG Information Society and Media, European Commission. 2007.
- [UNC03] UN/CEFACT Core Components Technical Specification, Version 2.0.1, 2003.
- [Wac03] Wache, H. Semantic Mediation for heterogeneous Information Sources (in German), Akademische V.-G. Aka. 2003.