Automating XML documents Transformations: A conceptual modelling based approach

A.Boukottaya[1], C.Vanoirbeek[1], F.Paganelli[2], O.Abou Khaled[3]

[1]Media Research Group
EPFL (Swiss Federal Institute of Technology)
1015 Lausanne, Switzerland

{aida.boukottaya, christine.vanoirbeek}@epfl.ch

[2]Electronics and Telecommunications Departement,
University of Florence, Italy

paganelli@achille.det.unifi.it

[3] Department of Computer Science, Engineering School of Fribourg, Switzerland

Omar.Aboukhaled@eif.ch

Abstract

The growing use of XML mark-up language has made a large amount of heterogeneous XML documents widely available. As the number of applications that utilize heterogeneous XML documents grows, the importance of XML documents transformations increases greatly. A serious obstacle for translating directly between two XML documents, using languages like XSLT, is that a mapping between the two XML representations needs to be carefully specified by a human expert. Current research attempts to address this problem by proposing algorithms to automate aspects of XML schemas matching task. In this paper, we identify two major problems encountered when current matching algorithms are used in the context of XML documents transformations. The first problem concerns possible scalability problem due to the diversity of schema constructs. The second problem deals with the need to perform semantic matching. We argue in favour of conceptual modelling as solution to avoid such problems. We introduce a new layered model for XML schemas, called Layered Interoperability Model for XML Schemas (LIMXS). LIMXS offers a semantic view for XML schemas through the specification of concepts and semantic relationships among them. We will show how our model transforms the matching algorithm into a dynamic and incremental algorithm that provide semantic mappings and helps to automate the transformation process.

Keywords: Automating XML documents transformations, Conceptual modelling, Layered Interoperability Model for XML Schemas, Semantic matching.

1 Introduction

1.1 Motivation

The growing use of XML mark-up language (XML 1.0 1998) has made a large amount of heterogeneous XML documents widely available. Heterogeneity arises in general from the fact that each organization or application creates its own documents structure according to specific requirements. These requirements are specified within a data model called *schema* (DTD or recently W3C XML schema standard (XML Schema 2001)) that describes XML document's valid content, and allowed structure. As the number of applications that utilize heterogeneous XML documents grows, the importance of XML documents transformations increases greatly. Currently, such transformations are manually encoded on a case-by-case basis using specific languages such as XSLT (XSLT 1.0 1999). XSLT, a recommendation of the World Wide Web Consortium, is a language, itself written in XML, with powerful computing capability encoding transformation of XML documents. An XSLT program (called stylesheet) is a set of template rules, each of which has two parts: a pattern that is matched against nodes in the source document and a template that can be instantiated to form the result document. The usual procedure to write such stylesheet requires an analysis of both the semantics and the structures of the source and target XML files to discover similarities between them. Manual coding then follows this analysis. However XSLT is a powerful transformation language, it has several drawbacks. It appears to be a complex language and simple transformations require the user to write a program, which needs complete mastery of XSLT language, and thus programming skills. Moreover, each time an XML document should be transformed to another XML document, a new XSLT program must be written. Several attempts at simpler and more formal transformation languages are underway but the issue of XML documents transformations remains complex.

1.2 Related Work

Currently, to perform XML document transformations, the burden falls on the human to first analyse both the semantics and the structures of the source and target XML documents, and second to manual coding the transformations. Many solutions have been proposed to simplify and automate XML documents transformations. Approaches can be distinguished along the following two dimensions: *Transformation specification* and *Schema matching*.

Transformation Specification

Several simpler and highly declarative transformation languages (S.Krishnamurthi, K.Gray, and P.Grauke 2000), (X.Tang and F. Tompa 2001) have been introduced as solutions to avoid programming. Special graphical tools have been also proposed to assist the specification of the transformations (E.Pietriga, J-Y.Vion-Dury, and V.Quint 2001), (XSLWIZ 2001). See (A. Vernet 2002) for more examples of transformation languages and tools. These languages and tools are very useful in describing and specifying transformations. However, they still require developers to manually indicate mappings for each source and target pair. Manual mapping is time consuming and thus especially unacceptable for applications where the information sources change frequently.

Schema Matching

A serious obstacle for translating directly between two XML schemas, using languages like XSLT, is that a mapping between the two XML representations needs to be carefully specified by a human expert. Since the XML schemas can be very diverse, the mappings created by the expert are often complex. This complexity makes them hard to maintain when original XML schemas change. An alternative strategy that is used for reconciling XML data is based on schema matching techniques to automate the mapping process (H.Su, H.Kuno, E.A.Rundensteiner 2001). Schema matching is the task of finding correspondences between two heterogeneous schemas. Schema matching is not a new problem and has been the focus of database community. (S.Castano, A. Ferrara, G. S. Kuruvilla , V.D.Antonellis 2002) describes the ARTEMIS tool environment to support the analysis and integration of sets of heterogeneous schemas by measuring the similarity of element names, data types and structures. (L. Xu, D.W. Embley, 2003) offers a novel integration approach that uses semi-automatic schema matching to produce source-to-target mappings. A recent survey of automatic schema matching (E. Rahm and P.A. Bernstein 2001) classifies approaches respecting to the *schemas information* (element naming, structure, data types, integrity constraints, etc.) and *auxiliary information* (generally domain specific common terminologies or thesaurus) used to discover schema similarities. However a lot of work has been done in this field, several issues remain unsolved when schema matching is used in the context of XML documents transformations. Such issues will be discussed in section 2.

1.3 Contribution of this paper

This paper aims at identifying encountered problems when current schema matching approaches are used in the context of XML documents transformations, and will introduce *conceptual modelling* as a solution to address these problems. The primary contributions of our work include:

- Unlike several approaches, we consider W3C XML Schemas and not DTDs. This makes the problem of XML schema matching more complex (we have to deal with features like typing, generalization/specialization, etc.)
- We introduce a *layered modelling approach* aiming to separate semantics from the syntactic nature of XML schemas. Semantics are represented using a conceptual model describing schema concepts and semantic relationships among concepts.
- Unlike current schema matching approaches, that generate numerical coefficients as schemas similarity measure, we will show how our layered modelling approach will enable the discovery of *semantic relationships* between schemas entities.
- We will introduce a *transformation framework* that encompasses the whole XML documents transformation process (modelling XML schemas, semantic matching, and transformation script generation).

The paper is organized in the following way. In section 2, we outline the limitation of current matching algorithms when applied in the context of XML documents transformations. Section 3 describes our proposed layered modelling approach for XML schemas. In Section 4 we present a taxonomy of XML Schemas heterogeneities and point a set of primitive transformation operations. Section 5 deals with the matching process. Section 6 gives an example that illustrates our approach. We describe our prototype system in section 7. Finally, we conclude the paper with a discussion and future work.

2 XML Schema matching problem

Recently, several schema matching techniques have been proposed. It is handled in some systems through machine learning techniques (A.Doan, P. Domingos, A.Y.Halevy 2001) to analyse data instances in order to predict element similarities. In other systems element- and structure-level matching is performed such as in (J. Madhavan, P.A. Bernstein and E. Rahm 2001) where authors combine linguistic and structural matching to predict element similarity. However a lot of effort has been done in the field of schema matching, two problems have to be addressed when schema matching is used for XML documents transformations applications: *scalability* and *Semantic relationships discovery*.

2.1 Scalability

Before performing schema matching task, schemas are always modelled in an abstract way that reflects their content structure and semantics. As example, simple labelled trees are generally used to describe DTDs. The matching process is strongly influenced by adopted *data model*.

A simpler data model with fewer data modelling constructs and constraints makes the task of matching easier, because it reduces conflict possibilities. As consequence, the transformation operations are simpler since the matching task involves fewer primitive operations. However, the adoption of such data models has several limitations in modelling information resources. One important restriction is that simple data models are generally poor in terms of types and conceptual abstractions. W3C XML Schemas add new features like user defined types and generalization / specialization relationships by means of substitution groups and subtyping mechanisms. To capture all XML schema features, we need a rich data model with complex modelling constructs and constraints. However the adoption of a complex data model present several advantages, it makes the task of matching complex. In (B. Omelayenko, and D. Fensel 2002) the authors prove that attempts to resolve all kind of conflicts (semantic conflicts, structural conflicts, etc.) within one transformation step generally causes a scalability problem. Such a solution would lead to complex set of mapping rules and consequently to possible performance problems.

2.2 Semantic relationships discovery

The key intuition of existing schema matching algorithms is to measure the similarity (linguistically and structurally) between *the labels* of elements. The similarity is generally expressed in term of coefficient in [0,1]. For example, the similarity coefficient between the two elements "*publication*" and "*book*" could be 0.8. When applied to XML documents transformations, we need to identify transformation operations (eg., *rename*, *merge*, *split*, etc.), this can not be done based on numerical coefficients. On the contrary, in order to infer adequate transformation operations, we need *semantic relationships* between schemas entities (eg., *equivalent*, *more general*, *incompatible*, etc.), that take into account data semantics. Semantics is the interpretation human attribute to data according to their understanding of the real world. For example if we know that "*book"* (from a source schema S1) is *equivalent* to *"publication"* (from a target schema S2)", instances of element "*book*" in S1 can be reused as instances for element "*publication*" in S2. To perform such matching, we need to map meanings of schema elements and not just their labels. An explicit and formal definition of the schema elements semantics and meaning is then required. Since XML Schema language, does not concentrate on the semantics of the content of XML documents, representing a *logical data model* that focuses on the syntactic structure of XML documents, we need to model XML schemas at *the semantic level* through a set of *concepts* and *semantic relationships*.

2.3 New requirements

Based on the above observations, to perform XML schema matching in the context of XML documents transformations, two new requirements for the matching algorithm have to be specified:

- First, we have to take into consideration all the features introduced by W3C XML schemas within the matching task. For this, we need a *rich data model* that express typing, and conceptual abstractions. This has to be done *without causing scalability problems*. Let us imagine that two applications need to exchange complex data. Instead of dealing directly with the details of semantics, structure and serialization of data, they can organize the data exchange in a layered fashion. This approach seems to be a good candidate to first ensure that the data model is enough rich to cover all features of XML schemas and second to avoid scalability problem. Layered modelling approach has been used in different contexts. In order to ensure the interoperability between web applications, (S. Melnik, S. Decker 2000) suggest a layered model ("divide to conquer" philosophy), presented as a series of three layers (syntax, object and semantic layers). Authors in (B. Omelayenko, and D. Fensel 2001) adopted a three layer modeling (syntax, data model and ontology) in order to reduce the complexity of information integration task.
- Second, the data model used to abstract the syntax of XML schemas has to give a semantic view that model XML schemas at *the semantic level* through a set of *concepts* and *semantic relationships*. Conceptual models provide a high level abstraction, being a good candidate to model XML schemas at semantic level. Conceptual modelling simplifies identification semantically related concepts because conceptual models have more clear semantics presenting *the real word view* independently of all implementation considerations.

3 Layered Interoperability Model for XML Schemas (LIMXS)

To reduce the complexity of the matching task and meet the new requirements introduced in section 2.3, we suggest a new data model for XML Schemas called LIMXS (Layered Interoperability Model for XML Schemas). LIMXS is essentially composed of two layers: *Semantic layer* and *Schema layer*, as shown in figure 1. The Semantic layer gives a way to semantically model XML Schema using a conceptual modelling approach. The schema layer is mainly concerned with detailed XML Schema constructs (elements/attributes declaration, simple/complex type definitions, constraint specification, etc). A one-to-one mapping between the two layers is also considered to map concepts on the semantic layer to the corresponding XML Schema constructs within the schema layer.

3.1 Semantic layer

The goal of this layer is to offer a *semantic view* of source and target schemas. A semantic view is the real word view described in terms of concepts and semantic relationships independently of all implementation considerations. The semantic layer describes *how data is structured* and *how data can be interpreted*. Several attempts have been proposed to represent XML schema at semantic level. (L.Feng, E. Chang, and T. Dillon 2002) proposes a semantic network, which provides semantic modeling of XML through a set of atomic and complex nodes, representing real world objects; a set of directed edges, representing semantic relationships between objects; a set of labels for nodes and edges; and finally, a set of constraints defined over nodes and edges . Authors in (N.

Routledge, L. Bird and A. Goodchild 2002) choose to conceptually modeling XML Schemas on the basis of the Unified Modeling Language (UML). An essential part of static UML is used to model XML Schemas. Other approaches (R. Xio, T. Dillon, E. Chang and L.Feng 2001) use object oriented methods to conceptually model XML Schemas. Our work is distinguished from the above ones in the following aspect. All these methods were used to offer a design methodology to write XML Schemas without exposing designers to low-level implementation details. As a first step a conceptual model or a semantic network is designed. Thanks to a set of mapping rules (eg. *each concept corresponds to an XML Schema complex type*), the XML serialization is then performed. In our work, we assume that there is no conceptual model available and re-engineering based on available logical XML schema will be necessary to obtain it. Partial inference of the data semantics is obtained from structures, namespaces, subtyping mechanisms, constraints, and substitution groups. System engineer intervention is then required to add semantics which can not be inferred from logical XML Schemas.

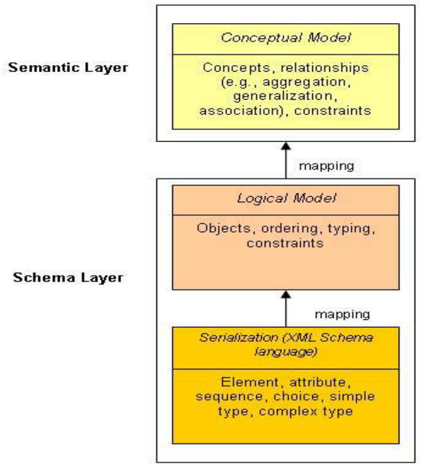

Semantic Layer
Conceptual Model
Concepts, relationships (e.g., aggregation, generalization, association), constraints
mapping
Schema Layer
Logical Model
Objects, ordering, typing, constraints
mapping
Serialization (XML Schema language)
Element, attribute, sequence, choice, simple type, complex type

Figure 1: Layered Interoperability Model for XML Schemas (LIMXS)

3.1.1 Conceptual Meta-model

The conceptual level is represented according to a *conceptual meta-model* that uses a subset of UML class diagram terminology (figure 2). A conceptual model consists of the following components:

- A set of **concepts**, representing real world objects sharing structure and semantics. Each concept has a name and an optionally set of properties.

- A set of **relationships**, representing *semantic relationships* between objects. We distinguish:

- *generalisation (specialisation):* It is a relationship between a more general concept C1 and a more specific one C2. It uses discriminator to show how concepts are extended (restricted). Generalisation/specialisation relationships are inferred thanks to XML Schema type extension/restriction mechanisms.
- *association:* is a semantic relationship between two concepts meaning that both are conceptually at the same level. Associations are inferred thanks to XML Schema substitution group mechanism and reference integrity definitions (Key/Keyref).
- *aggregation:* is a composition (part-whole) relationship between two concepts. Aggregation relationships are inferred from hierarchical structure expressed by means of "sequence", "choice" and "all" constructs at schema level.

- A set of **constraints**, defining multiplicity constraints over concepts and relationships.

3.1.2 Conceptual Model Re-engineering

Conceptual model re-engineering is a bottom up process aiming to analyse logical XML Schemas and default semantics that they carry in order to obtain a conceptual model valid against the proposed meta model. We propose a set of rules based on default semantic interpretation of XML Schema constructs. These rules define the correspondences between schema constructs and conceptual meta-model constructs. We take into consideration relevant XML Schemas features such as Complex types, unnamed types, abstract types, etc. Two distinct rules categories are defined according to the nature of constructs involved in the rule.

Concept discovery Rules: are rules related to the discovery of concepts.

Relation discovery Rules: are rules related with the discovery of relationships between concepts and their semantics.

We do not further detail the process of conceptual model re-engineering in this paper. Figure 3 introduces examples related to concept discovery rules and relation discovery rules.

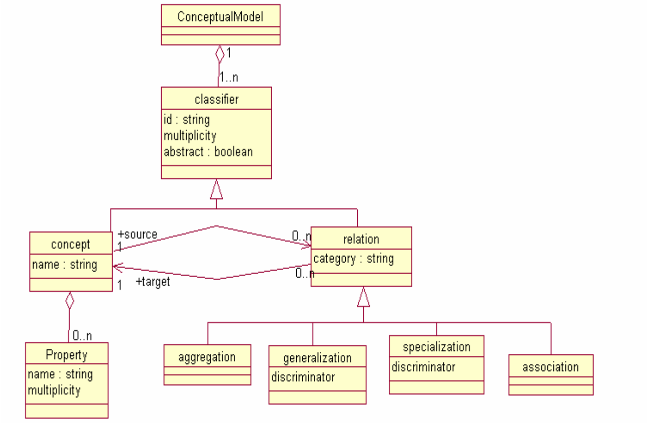
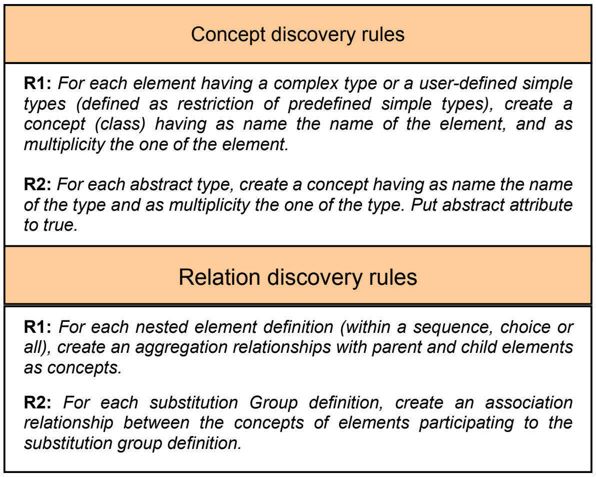


Figure 2: Conceptual Meta Model

3.2 Schema Layer

To overcome the limitations of DTDs in modelling XML data, several languages, called *schema languages*, were introduced for describing XML documents logical structure.

(D. Lee, W. Chu 2000) proposes a comparative analysis of such schema languages. In our work, we were first interested on W3C schema language features, but we notice that other proposed schema languages share the same features. As example, DSD 1.0 (N. Klarlund, A. Moller, M. I. Schwatzbach 2000) introduces user defined types, unordered sequences, etc. SOX 2.0 (A. Davidson, M. Fuchs, M. Hedin 1999) allows simple types restriction and complex types extension. However, these languages share some common features, each schema language offers a set of constructs to describe XML information (e.g., W3C XML Schema uses a grouping construct ⟨all⟩ to specify the unordered sequence. In (C. Frankston, H. S. Thompson 1998), the ⟨order='many'⟩ attribute specifies that sub-elements can appear in any order). However, they can share the same features; these schemas languages are heterogeneous at syntactic level. To take into consideration such languages within our work, the schema layer is composed of two sub layers: *the logical layer* and *the serialisation layer*. The aim of the logical layer is to describe different schema languages constructs independently of syntactic details.



Concept discovery rules

R1: *For each element having a complex type or a user-defined simple types (defined as restriction of predefined simple types), create a concept (class) having as name the name of the element, and as multiplicity the one of the element.*

R2: *For each abstract type, create a concept having as name the name of the type and as multiplicity the one of the type. Put abstract attribute to true.*

Relation discovery rules

R1: *For each nested element definition (within a sequence, choice or all), create an aggregation relationships with parent and child elements as concepts.*

R2: *For each substitution Group definition, create an association relationship between the concepts of elements participating to the substitution group definition.*

Figure 3: Example of Rules for Conceptual Model Re-engineering

3.2.1 The logical layer

The logical level presents in an *abstract* and *graphical* way how XML schema constructs are combined to describe the information of the semantic layer. Features like ordering relationships, typing, and integrity constraints are described. The logical layer gives an object oriented view of the information independently of the used XML schema language. Figure 4 gives an example showing how simple XML schema types are represented at the logical level. To each simple type, corresponds a *basic object* having a unique ID. Each basic object has a basic content and a set of associated constraints. A basic content can be:

- an atomic value: a value from the domains of basic data types (e.g., string, integer, date, etc.).

- a constructional value: which can be a list value or a union value. A list value is an ordered collection of items having the same basic content. A union value allows any of members of a collection as the returning value.

Constraints related to basic objects are uniqueness constraints, referential integrity constraints and domain constraints (e.g., restrict the range of numerical values by giving the maximal/minimal values).

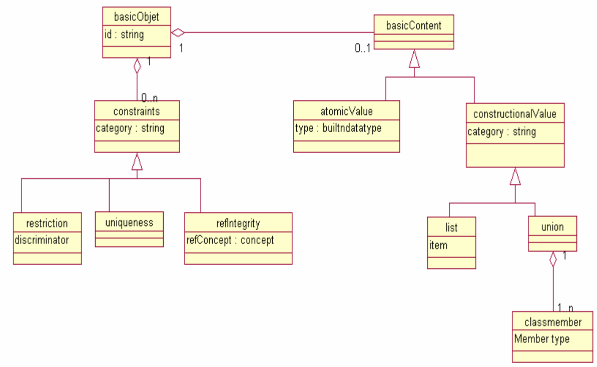


Figure 4: Logical Meta Model

3.2.2 The serialisation layer

This layer describes a *specific implementation* of data structures described in the logical model. It depends on a particular product or version. The serialization layer, in our context, uses the standard textual language defined by the World Wide Web (XML Schema Recommendation) or any other XML schema language.

In next sections, we will show how our layered model (LIMXS) simplifies XML documents transformation process and improve the quality of matching results.

4 Transformation Operations

Based on the LIMXS model, we identify causes of heterogeneity between XML schema entities at different levels. We classify these causes into two categories: *semantic heterogeneities*, and *logical heterogeneities*.

4.1 Semantic heterogeneities

The distinction between two schemas is done based on the comparison of their respective semantic views. Semantic heterogeneities deal essentially with differences in structure (*how are the data organized?*) and differences in interpretation (*what do the data mean?*). For instance, different names (eg. *author* and *writer*) can be attached to the same concept in the two schemas. Let us imagine two schemas that represent information about *persons* and their emails. In the first schema information is modeled by means of two aggregation relationships. The first relates each *person* to his *mailbox* and the second relates a *mailbox* to a set of *mails*. While in the second schema, only one relationship between *person* and *mail* is used.

Figure 5 gives further examples of semantic heterogeneities.

4.2 Logical heterogeneities

The richness of XML schema languages gives rise to a larger variety of possibilities to model the same concepts. For example, *dates* may be represented as *strings* in one schema, or in another schema as instances of the primitive type "*Date*". These conflicts are very difficult to solve, and require in general human intervention. In this case the mapping process provides several suggestions to the user based on some heuristics and a predefined library of logical transformation operations, so that the mapping process can continue. Figure 6 gives further examples of logical heterogeneities.

4.3 Taxonomy of transformation operations

According to the causes of heterogeneities described in the previous section, we propose a set of primitive transformation operations that are the building blocks that would enable schemas transformation. These primitive operations can be composed together to represent larger transformations. Moreover, we provide a library of predefined operations in order to deal with logical heterogeneities.

4.3.1 Conceptual primitive operations

Conceptual primitive operations concern concepts and relationships at the semantic layer. These operations include:

- *Add*: add an entity to the target schema. Entities can be relationships, concepts and properties.
- *Delete*: carries out the opposite transformation.
- *Merge:* Two distinct entities are merged into one entity. Concepts like *street*, *country* and *state* can be separate in source schema and merged as *address* concept in target schema. Then the values of *address* in the target schema match a concatenation of values from source schema concepts.
- *Split*: This is the reverse operation of merge. The values of target schema match then a decomposition of a source schema concept.
- *Rename*: change concept and properties names.
- *Connect:* The connect operation is one to one mapping that maps two equivalent entities without any transformation.

4.3.2 Logical operations

Logical heterogeneities are very difficult to resolve since they require extra information that only the user can provide. By analysing XML Schemas, we provide a set of common functions within a library that the user can easily modify or extend. Examples of such functions deal with type compatibility problem. Imagine that the target type is included in the source type. We suggest functions like *rounding a real to an integer*, *truncating string to a given length*, etc. Logical operations deal also with differences in schema constraints such as cardinality, default and Null values.

5 Layered matching approach

Map is defined as a schema manipulation operation that takes in input two heterogeneous schemas and returns a mapping that identifies corresponding entities in the two schemas. We propose a *layered mapping approach* according to our interoperability model. Imagine two XML enabled applications that need to exchange data. Instead of creating in a static manner a document specifying correspondences between schema entities, we propose a *dynamic* and *incremental* mapping process outlined in figure 7. Semantic matching aims to discover similarities expressed in term of *semantic relationships* between a source and target semantic views. Once semantic mapping is generated and validated by human intervention, the result is given to the logical layer, which performs matching at logical level. Matching at this level focuses on discovering and resolving logical conflicts such as *representation differences* (use of ordered lists (sequences) Vs choices), *datatype conflicts*, and *constraints conflicts*. Finally, a transformation script can be automatically produced. The remaining of this section describes in detail the semantic matching algorithm.

Semantic conflicts

Different structures

- Single concept Vs merged concepts (Title Vs ShortTitle & extendedTitle)
- Different granularity (course/teacher Vs course/department/teacher)
- Aggregation conflicts (PersonalInfo as name & e-mail Vs PersonalInfo as name & address & e-mail)
- Generalization conflicts (separate concepts assistant & teacher Vs a generalized concepts CourseResponsible and two subconcepts assistant & teacher)

Different naming (Author Vs Writer)

Figure 5: Semantic heterogeneities

Logical conflicts

Conflicts between model constructs:

Different representations (ordered list "Sequence" Vs unordered list "All")

Different Types

- One element has simple type, the other has complex type
- Different Types (date Vs string)
- One type is defined with facets, the other is a predefined type

Different constraints (default values, Null values, Cardinality)

Figure 6: Logical heterogeneities

Semantic matching

So far, with the exception of (L.Serafini, P.Bouquet, B.Magnini and S. Zanobini 2003), none of the current matching approaches perform semantic matching. However authors outline the need to perform semantic matching, they restrict they work on hierarchical classifications (supporting only {is-a} relationships). The validity of the generated mappings is ensured by testing propositional satisfiability. Iterations are needed when matching results are not good. In our work, we do not restrict our selves to hierarchical classifications, on the contrary semantic matching involves rich conceptual models (with aggregation, association, generalization and specialization relationships). We do not use propositional satisfiability approach, but we infer semantic relationships based on the richness of conceptual model structures. Specific user input is interactively requested at critical points, and not just at post matching. This makes post matching easier, since not good matches are corrected while running the matching process and do not propagate. The key idea is that in semantic matching we explore the fact that schema semantics are explicitly described through concepts and semantic relationships. The result of semantic matching is a set of semantic mappings. We adopted a two phases (linguistic analysis and structural analysis) process that extracts semantic relationships between concepts.

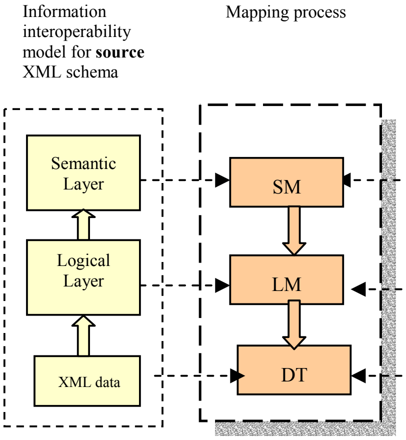

Information interoperability model for **source** XML schema
Mapping process
Information interoperability model for **target** XML schema
Semantic Layer
Logical Layer
XML data
SM
LM
DT
Semantic Layer
Logical Layer
XML data

Figure 7: The Three layered Matching approach

SM: Semantic matching
LM: Logical matching based on the previous generated semantic matching)
DT: Data transformation: generation and execution of XSLT code

Linguistic analysis

The first phase focuses on a *linguistic analysis* of concept meanings. We use WordNet (A.G. Miller 1995), an electronic lexical database where relations like homonymy, synonymy are available to relate word meanings. By querying WordNet, we are able to define semantic relationships between concepts. We consider five semantic relationships including *equivalent*, *general, specific*, *compatible* and *incompatible*. For example two concepts are equivalent if they are synonyms (e.g., Authors and Writer concepts).

Structural Analysis

The aim of the structural analysis is to extend semantic relationships discovered by linguistic analysis by examinating the organisation of data provided by the semantic views. The organisation of data is described through aggregation, association and generalisation relationships. Let us consider a simple example, WordNet querying gives that a source concept “book” is a specific concept of a more general target concept “publication”. If a source semantic view specifies that “monograph” is a restriction of “book”, then a semantic relationship “monograph is a specialisation of publication” can be inferred. Structural analysis involves several steps including aggregation analysis, generalisation analysis, specification analysis and association analysis.

Semantic matching Result

Semantic matching gives as output a set of semantic mapping elements. The key feature of our work is to provide a layered modelling approach that includes modeling of mappings and transformations. For modeling mappings and translation of a source XML schema to a target XML schema, we use (1) the information interoperability model, (2) primitive transformation operations and (3) the library of transformation functions. A semantic mapping M is then a 6-Tuple **<M$_{id}$, T, SE, TE, SR, COP, RM>** where:

- **Mid** is a unique identifier of the given mapping element.
- **T** is the type of the mapping element. Based on the conceptual meta model described in section 3.1.1, three types of mappings are considered. (1)*concept mapping* (relating concepts), (2) *property mapping* (relating properties), and (3) *relation mapping* (relating relations). See section 6 for a detailed example.
- **SE** is a set of source entities (concepts, relations, and properties) involved in the mapping.
- **TE** is a set of target entities (concepts, relations, and properties) involved in the mapping. We consider OneToOneMapping, OneToManyMapping, ManyToOneMapping and ManyToManyMapping.
- **SR** is a set of semantic relationships between entities being mapped.
- **COP** is a set of conceptual operations (defined in section 4.2.1). Each operation has a name and a set of constraints that represent the conditions that should be verified in order to execute the operation.
- **RM** (*Related mappings)*: this reflects how mappings are related and may be combined into more complex mappings. Several relations may hold between mappings:
    - *Composition:* is a relation specifying that a mapping is composed of other mappings. Example in section 6 illustrates a composition relationship

between concept mapping “map11” and attribute mapping “map111”.

- *Generalization/specialization*: allows one mapping to reuse definitions from another mapping and respectively to extend or restrict those definitions. Example in section 6 illustrates a generalization relationship between concept mappings “map11” and “map2”.

6 Example

Let us consider Figure 8 where parts of two XML schemas semantic views (conceptual models) are represented. Both conceptual models are taken from the university domain to keep the example simple.

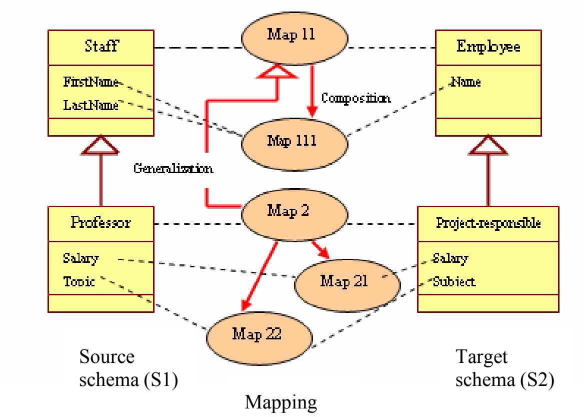


Figure 8: Example of generated mappings and relations between mappings

We first begin to perform semantic matching. By querying WordNet, we generate a semantic relationship (0) *concepts “staff” and “employee” are semantically equivalent*. A first mapping, having “mapp11” as ID, is then set between *S1.staff* and *S2.employee*. Properties of these concepts must be further mapped. *S1.staff* distinguishes between “*First name*” and “*Last name*” while *S2.employee* does not. Linguistic analysis gives a first semantic relationship (1) “*First name*” and “*Last name*” are specifications of “name”. Structural analysis extracts semantic relationships (2) firstname *is-property* of staff, (3)lastname *is-property* of staff, (4) name *is-property* of employee. (0), (1), (2), (3),and (4) allow the system to suggest that the concatenation of “*First name and Last name*” and *“name”* are *equivalent.* Interactive user validation at this point, allows to create ManytoOne mapping (ID=map111) with *merge* transformation operation.

```
<Mapping ID = “map1”>
  <SourceSchema  source=”S1.xsd”/>
  <TargetSchema   target=”S2.xsd”/>
  <HasMappings  OneToOnemapping ID=”map11”/>
</Mapping>
```

```
<OneToOnemapping ID=”map11”>
  <Conceptmapping>
     <SourceConcept concept=”Staff”/>
     <TargetConcept concept=”employee”/>
     <Transformation>
       <Operation name=”rename”/>
     </transformation>
     <HasMappings  ManyToOnemapping ID=”map111”/>
  <Conceptmapping/>
<OneToOnemapping/>
```

```
<ManyToOnemapping ID=”map111”>
  <Attributemapping>
     <SourceAttribute attribute=”FirstName”/>
     <SourceAttribute attribute=”LastName”/>
     <TargetAttribute attribute=”Name”/>
    <Transformation>
       <Operation name=”Merge”/>
     </transformation>
 <Attributemapping/>
<ManyToOnemapping/>
```

The semantic matching process will try then to map concepts *S1.professor* and *S2.project-responsible*, since structural analysis shows that concepts *S1.professor* and *S2.project-responsible* are *sub concepts* respectively of already *equivalent* concepts *S1.staff* and *S2.employee*. At this level a first suggestion is made by the system: *S1.professor* and *S2.project-responsible are equivalent.* Properties related to both concepts are then analysed. A shared property “salary” is found and linguistic analysis (WordNet) confirms that properties “topic” and “subject” are equivalent. The system can then conclude that *S1.professor* and *S2.project-responsible are equivalent*. A semantic mapping (map 2) is generated as sub mapping of map11 (mapping between *S1.staff* and *S2.employee*).

Map21 and map22 are respectively mappings between *S1.professor*’s property “*salary*” and *S2.project-responsible*’s property “*salary*” and *S1.professor*’s property “*topic*” and *S2.project-responsible*’s property “subject”. The transformation operations required are respectively *connect* and *rename*.

```
<OneToOnemapping ID=”map2” extendmapping=”map11”>
  <Conceptmapping>
     <SourceConcept concept=”professor”/>
     <TargetConcept concept=”project-
                             responsible”/>
     <Transformation>
       <Operation name=”rename”/>
     </transformation>
     <HasMappings  OneToOnemapping ID=”map21”
                   OneToOnemapping ID=”map22”/>
  <Conceptmapping/>
<OneToOnemapping/>
```

```
<OneToOnemapping ID=”map21”>
  <Attributemapping>
     <SourceAttribute attribute=”salary”/>
     <TargetAttribute attribute=”salary”/>
    <Transformation>
       <Operation name=”connect”/>
     </transformation>
 <Attributemapping/>
<OneToOnemapping/>
```

```
<OneToOnemapping ID=”map22”>
  <Attributemapping>
     <SourceAttribute attribute=”topic”/>
     <TargetAttribute attribute=”subject”/>
    <Transformation>
       <Operation name=”rename”/>
     </transformation>
 <Attributemapping/>
<OneToOnemapping/>
```

Once semantic mappings are generated. The system deals with logical matching. Imagine that property *salary* in the source schema S1 is of type “*Real*”, while the property *salary* in the target schema S2 is of type “*integer*”. In this case the semantic mapping has to be augmented by a new transformation operation that round a *real* to an *integer*. An integer rounded from a real approximate the real but also all others that round the same integer. Before executing such transformation, user validation is needed. If the user has another suggestion like changing type integer to real in the target schema, the round operation will not occur otherwise the mapping “map21” will be incremented by a new operation “*roundrealtointeger*”.

```
<OneToOnemapping ID=”map21”>
  <Attributemapping>
     <SourceAttribute attribute=”salary”/>
     <TargetAttribute attribute=”salary”/>
    <Transformation>
       <Operation name=”connect”/>
       <Operation name=”roundrealtointeger”/>
     </transformation>
 <Attributemapping/>
<OneToOnemapping/>
```

7 The Prototype System

The prototype system, we are developing, incorporating all the ideas discussed in the paper, consists of three *transformation modules: Conceptualization toolkit, Matcher engine, and execution engine*. Figure 9 outlines the whole system architecture.

Conceptualization toolkit

For the generation of semantic and logical views from XML schemas, we aim to develop a *conceptualization toolkit* that will be composed essentially of two graphical tools: *Semantic view editor and viewer*, *Logical view editor and viewer*. These graphical tools will encourage the user to either add semantics or extend the generated views.

Matcher engine

We develop a *Matcher engine* to perform semantic and logical matching. The matcher engine uses additional modules: an interface for querying WordNet and a graphical user interface that allows the users to validate mappings generated by the system, and provide further domain constraints. We also provide a mapping evolution module. This module focuses on storing the generated mapping and on their synchronization with the changes in source and target schemas. Keeping mappings evolution allows their reusability. The goal is to avoid reapplying the mapping process every time schemas change.

Execution engine

The execution engine is responsible for parsing mapping results and generating automatically XSLT transformation scripts. It also permits the translation of data instances (XML files) valid against a source schema to instances valid against a target schema.

8 Conclusion and Future Work

Due to the extensive use of XML markup language in several domains, there has been a great interest on proposing rich data models (XMLSchemas) that reflects document semantics and structure. The existence of such rich schemas has made a large amount of heterogeneously XML documents widely available. In this framework, XML documents transformations are of major concern. Currently, to perform XML documents transformations, the burden falls on the human to first analyse both the semantics and the structure of the source and target XML documents, and second to manual coding the transformations using specific languages such as XSLT.

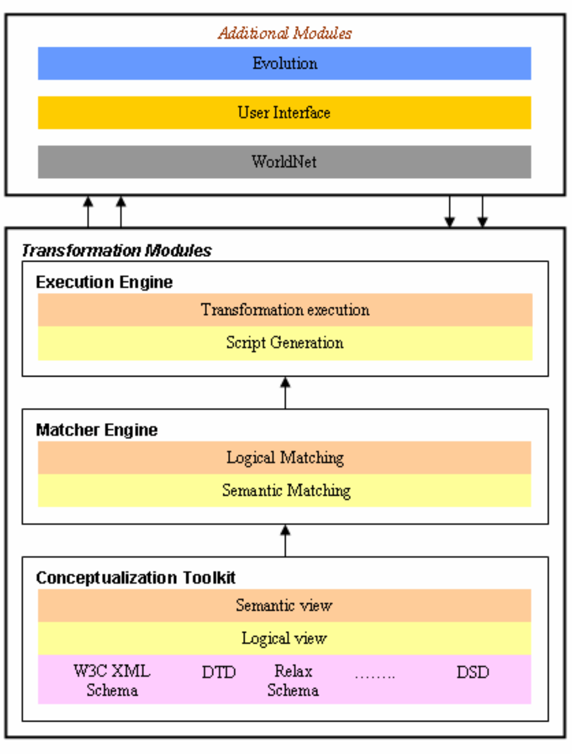


Figure 9: Prototype architecture

Many solutions have been proposed to simplify and automate XML documents transformations, but none really encompass the over all process. This work proposes an approach for automating the transformation of XML documents. We have specially focused on two fundamental problems: *How to deal with all features of*

XML schema without causing a scalability problem and *how to discover semantic relationships between schema entities.* For this end, we propose a layered data model that we call *Layered Interoperability Model for XML Schemas (LIMXS).* The proposed Model has two major advantages. In one hand, it is a rich model able to model all XML Schemas features such as typing, specialization/generalization relationships, namespaces, etc. In the other hand, it offers a semantic view and a logical view of XML schemas based on conceptual modelling. The semantic view described in term of concepts and semantic relationships gives an explicit and formal definition of the schema elements semantics and meaning, which allows us to map meanings of schema elements and not just their labels. The logical view describes schema constructs independently of syntactic details, and allows us to perform matching at logical level.

In the future, we intend first to finalise the prototype system and especially the conceptualization toolkit and the execution engine. We also plan to design an evaluation technique for quantifying XML Schema transformations based on cost of transformation operations in the case where multiple transformation strategies are possible.

9 References

XML 1.0. (1998) Extensible Markup Language (XML) Version 1.0. World Wide Web Consortium. http://www.w3.org/TR/REC-xml.

XML Schema. (2001) W3C Recommendation, "XML Schema Primer", W3 Consortium, available at http://www.w3.org/TR/xmlschema-0, 2001.

XSLT 1.0. (1999) W3C Recommendation. XSL Transformations XSLT Version 1.0, Available at http://www.w3.org/TR/xslt (June 2002).

S.Krishnamurthi, K.Gray, and P.Grauke. (2000) Transformation-by-example for XML. *Proceeding of the Second International Workshop on Practical Aspects of Declarative Languages (PADL'00),* Lecture Notes in Computer Science, Vol. 1735, pages 249-262.

X.Tang and F. Tompa. (2001). Specifying transformations for structured documents. *In proceeding of 4th International Worshop on Web and Databases (WebDB 2001),* Pages 67-72.

E.Pietriga, J-Y.Vion-Dury, and V.Quint.(2001). Vxt: a visual approach to XML transformations. *Proceeding of the ACM Symposium On Document Engineerin.*

XSLWIZ. (2001). http://www.induslogic.com/products/xslwiz.html

A. Vernet. (2002) XML transformation languages. http://www.scdi.org/~avernet/misc/xml-transformation

H.Su, H.Kuno, E.A.Rundensteiner. (2001) Automating the transformation of XML Documents. *Proceeding of the ACM Symposium On Document Engineering.*

Silvana Castano, Alfio Ferrara, G. S. Kuruvilla Ottathycal, Valeria De Antonellis (2002): A Disciplined Approach for the Integration of Heterogeneous XML Datasources. DEXA Workshops 2002: 103-110

Li Xu, David W. Embley: Discovering Direct and Indirect Matches for Schema Elements. DASFAA 2003: 39-46

E. Rahm and P.A. Bernstein. (2001). On Matching Schemas Automatically, *Technical Report, Dep. Of Comp science, Univ of Leipzig.*

A.Doan, P. Domingos, A.Y.Halevy (2001). Reconciling schemas of disparate data sources: A machine learning approach. *In SIGMOD'01.*

J. Madhavan, P.A. Bernstein and E. Rahm (2001). Generic Schema matching with Cupid. *Proc. 27 th Int. Conf. On Very Large Data Bases (VLDB).*

B. Omelayenko, and D. Fensel (2002). Analysis of B2B Catalogue Integration Problems. Kluwer Academic Publisher. 270-277.

S. Melnik, S. Decker (2000). A Layered Approach To Information Modelling and Interoperability *On The Web. Proc. Of the workshop on the semantic Web at the 4 th European Conference on Research and Advanced Technology for Digital Librrries ECDL.*

B. Omelayenko, and D. Fensel (2001) Scalable Document Integration for B2B Electronic Commerce. http://citeseer.nj.nec.com/452604.html

L.Feng, E. Chang, and T. Dillon (2002). A Semantic Network- Based Design Methodology for XML Documents. *ACM Transactions on Information Systems (TOIS) Volume 20 , Issue 4. Pages: 390 – 421.*

N. Routledge, L. Bird and A. Goodchild (2002). "UML and XML Schema", *ADC'2002.*

R. Xio, T. Dillon, E. Chang and L.Feng (2001). Modeling and Transformation of Object Oriented Conceptual Models into XML Schema. *DEXA 2001, LNCS 2113, Pages795-804.*

D. Lee, W. Chu (2000). Comparative analysis of six {XML} schema languages. *SIGMOD Record (ACM Special Interest Group on Management of Data), Vol 29, num 3, pages 76-87.*

N. Klarlund, A. Moller, M. I. Schwatzbach, (2000). ``DSD: A Schema Language for XML", *Proc. 3rd ACM Workshop on Formal Methods in Software Practice.*

A. Davidson, M. Fuchs, M. Hedin, (1999) ``Schema for Object-Oriented XML 2.0", *W3C.* (http://www.w3.org/TR/NOTE-SOX)

C. Frankston, H. S. Thompson. (1998) ``XML-Data reduced", *Internet Document.* (http://www.ltg.ed.ac.uk/~ht/XMLData-Reduced.htm)

L.Serafini, P.Bouquet, B.Magnini and S. Zanobini (2003).An Algorithm for Matching Contextualized Schemas via SAT. *IRST Technical Report 0301-06, Instituto Trentino di Cultura.*

A.G. Miller (1995). WordNet: A lexical Database for English. ACM 38 (11). Pages 39-41.