Using XSEM for Modeling XML Interfaces of Services in SOA*

Martin Nečaský

Department of Software Engineering, Faculty of Mathematics and Physics, Charles University
martin.necasky@mff.cuni.cz, http://www.necasky.net

Abstract. In this paper we briefly describe a new conceptual model for XML data called *XSEM* and how to use it for modeling XML interfaces of services in service oriented architecture (SOA). The model is a combination of several approaches in the area of conceptual modeling of XML data. It divides the process of conceptual modeling of XML data to two levels. The first level consists of designing an overall non-hierarchical conceptual schema of the domain. The second level consists of deriving different hierarchical representations of parts of the overall conceptual schema using transformation operators. Each hierarchical representation models an XML schema describing the structure of the data exchanged between a service interface and external services.

Keywords: conceptual modeling, XML, XML Schema, SOA

1 Introduction and Motivation

Recently, XML has been used for an exchange of data between heterogeneous information systems, for an internal data representation, and also as a logical database model. Therefore, modeling of XML data should become an inseparable part of the application data modeling process on the conceptual level.

For example, assume a medical application integrating data about patients from several external sources. The application works as a service. It is a black box that stores the patient data in an internal database in an internal representation and provides access to the database through predefined interfaces used by external services such as hospital systems or insurance systems. The data exchanged between an external service and the medical service is in an XML form. Each interface provides an XML schema describing the form in which the data is presented to and received from the external services through the interface. The external services do not know the structure of the internal database. They only know the XML schemes provided by the interfaces.

* This paper was supported by the National programme of research (Information society project 1ET100300419)

J. Pokorný, V. Snášel, K. Richta (Eds.): Dateso 2007, pp. 35–46, ISBN 80-7378-002-X.

Assume an interface I_{exam} for exchanging results of medical examinations and an interface I_{diag} for exchanging medical diagnoses in XML. Both interfaces define XML schemes describing the required structure of exchanged XML documents. We can imagine the following scenario:

1. physician in a hospital makes a diagnosis of a patient; to decide the diagnosis, the physician needs the results of a patient's examination performed in a different hospital
2. physician requests the hospital system for the results; hospital system requests the medical service for the results through I_{exam}
3. medical service exports the results from the internal representation into an XML document with the structure defined by I_{exam} and sends it back to the hospital system
4. hospital system receives the XML document and presents the data to the physician; physician diagnoses a patient's disease and records the diagnosis to the hospital system
5. hospital system exports the diagnosis data into an XML document with the structure defined by I_{diag} and sends it to the medical service through I_{diag}
6. medical service receives the XML document with the diagnosis and stores the data into the internal database

Fig. 1 shows how the medical service similar to our example would be organized today. The figure shows the internal structure of the service. There is the internal database and a conceptual schema describing the structure of the database. For the external hospital system, the service is a black box. The hospital system communicates with the service through the interfaces I_{exam} and I_{diag}. The figure illustrates the scenario described above.

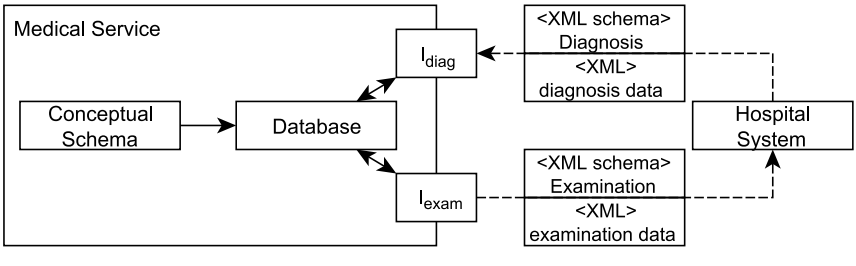

Fig. 1. Motivation

The connection of the example with the conceptual modeling is that there is a need to model the structure and semantics of XML documents exchanged between the medical service and external services/applications through the interfaces on the conceptual level. However, the following problems can occur:

1. conceptual schema and the XML schemes describing the structure of the data exchanged through the interfaces can be missing
2. if the conceptual schema and the XML schemes are present, there is no explicit binding between them (i.e. the XML schemes have to be created and maintained manually)
3. scripts for extracting data from the XML documents and transforming the data into the internal representation and vice versa must be created and maintained manually

The challenge is to eliminate these problems (1-3) by introducing a conceptual model for XML data. Such a model must allow to design an overall conceptual schema of the service domain and to derive the XML schemes describing the service interfaces (1). Even though the XML schemes organize the data into hierarchies, the overall conceptual schema need not be hierarchical. The explicit binding between the XML schemes and the overall non-hierarchical conceptual schema facilitates the maintenance of the XML schemes and the creation and maintenance of the scripts that transform data between the internal representation and the interface XML representations (2,3).

2 Related Work

If we want to model XML data on the conceptual level, we have to deal with some special features of XML data such as irregular structure, ordering, mixed content, and a hierarchical structure.

There are some approaches, for example ERX [6] or XER [8], extending the E-R model to be suitable for the conceptual modeling of XML data. Because E-R is not hierarchical (there are $M : N$ relationship types and n-ary relationship types), XML schemes must be derived in some way. The problem is that a user can not specify how the data should be organized in hierarchical XML. The hierarchical structure is derived automatically without following user requirements.

Another possibility of how to model XML data is to start from a hierarchical structure. This approach may be called *hierarchical approach*. There are conceptual models based on the hierarchical approach, for example ORA-SS [1]. Using this approach we can model hierarchical structures easily. However, a problem with modeling of non-hierarchical structures arises. Moreover, hierarchical schemes are not so transparent as non-hierarchical E-R schemes. A designer must think about the data in hierarchies which is not natural in general.

The problem of the approaches is that it is not possible to design one or more hierarchical organizations of parts of an overall non-hierarchical conceptual schema of the domain as it is required by the example medical service. Moreover, it is not possible to derive different hierarchical organizations of the same parts of the overall conceptual schema.

We propose a new conceptual model for XML called *XSEM* trying to solve the mentioned problems. In [4], we offer a survey of conceptual modeling for

XML. We propose a detailed list of requirements for conceptual models for XML, describe recent conceptual models for XML in a unified formalism, and compare the described models against the requirements. In [5], we describe our *XSEM* in a formal way.

In this paper we describe XSEM briefly and we show its possible application to modeling of XML interfaces of SOA services. There are two contributions of this paper. First, we show how XSEM can be applied to conceptual modeling of XML interfaces of SOA services and how it can facilitate important processes in the service creation and maintenance. Second, because we show a practical application of XSEM, we also concentrate on presentation features of XSEM. We show that it is necessary to extend the XSEM constructs proposed in [5] to present XSEM schemes in a transparent way.

3 Idea

We illustrate our idea with the architecture of the medical service. There is the internal logical database schema describing the structure of the data stored in the internal database. The medical service provides several interfaces used by external applications to access the internal database. Each interface provides an XML schema describing the structure of XML documents that are exchanged between the service and the external applications through the interface. We can comprehend the XML schemes as hierarchical views on parts of the internal logical schema. Each group of external applications needs different structure of XML documents. These documents can contain the same data, but in different hierarchical organizations.

Following the architecture of the medical service, we need to design an overall non-hierarchical conceptual schema of the domain. From the overall schema, we need to derive several hierarchical conceptual views. These views describe the XML schemes for the interfaces. The derivation must be driven by a designer. The hierarchical view design process consist of selecting the components of the overall schema that should be represented in the view followed by the specification of how the components should be organized in the hierarchy. At the end, the XML schemes are derived automatically from the conceptual hierarchical views.

4 XSEM Model

XSEM is a conceptual model for XML based on the previous idea. It divides the conceptual modeling process to two levels. On the first level, we design an overall non-hierarchical conceptual schema of our domain using a part of XSEM called *XSEM-ER*. On the second level, we design hierarchical conceptual schemes using a part of XSEM called *XSEM-H*. XSEM-H schemes are not designed separately from the XSEM-ER schema. We derive them from the XSEM-ER schema by so called *transformation operators*. Each XSEM-H schema is a hierarchical view on a part of the XSEM-ER schema. It describes required XML schema on

the conceptual level and there is an explicit binding between the hierarchical organization and the semantics of its components.

We can easily apply XSEM to model XML interfaces of SOA services. First, we design an overall XSEM-ER schema of the service domain and then we derive an XSEM-H schema for each service interface. The XSEM-H schema describes the XML schema for the XML data exchanged through the interface.

4.1 XSEM-ER

XSEM-ER is an extension of E-R proposed by Chen. It allows to model the special XML features like irregular structure, ordering, and mixed content. On this level, it is not important how the modeled data is organized in hierarchies. These hierarchical organizations are derived during the second part of the modeling process.

Fig. 2 shows an example XSEM-ER schema modeling a small part of the medical domain. As in the classical E-R model, there are strong and weak entity types and relationship types. *Strong entity types* represent stand alone objects and are displayed as boxes. Fore example, there is a strong entity type *Hospital* modeling hospitals or a strong entity type *Physician* modeling physicians. *Weak entity types* represent objects that depend on another objects. We display them as boxes with an inner hexagon. Each entity type the weak entity type depends on is connected with the box by a solid arrow. We call these entity types as *components* of the weak entity type. For example, there is a weak entity type *Department* with a component *Hospital* modeling departments of hospitals. *Relationship types* represent relationships between objects. Therefore, a relationship type is composed of one or more entity types called *components* of the relationship type. We display relationship types by hexagons connected by solid arrows with their components. For example, there is a binary relationship type *Employ* that represents employments of physicians at departments of hospitals or at separate clinics (an extending modeling construct described in the following text is used to unify departments and clinics).

XSEM-ER adds new modeling constructs called *data node types*, and *outgoing* and *incoming cluster types* for modeling special XML features. A *data node type* is connected with an entity type which is called *component* of the data node type. It represents data values assigned to an instance of the component. These values are not attribute values of the entity. They are data values which are mixed with the relationships and weak entities having the entity as a component value. In a graphical representation, a data node type is displayed as an ellipse with a name of the data node type.

For example, assume a patient visiting a physician (represented by the weak entity type *Visit* at Fig. 2). During the visit, the physician writes a description of the course of the visit. The description is not an attribute value of the visit, it is an unstructured text assigned to the visit. Moreover, the text is mixed with the examinations made during the visit. At Fig. 2, we use a data node type *VTxt* and an incoming cluster type (see the following text) to model this situation. Ex. 1 is a motivating example for the introduction of data node types. There is

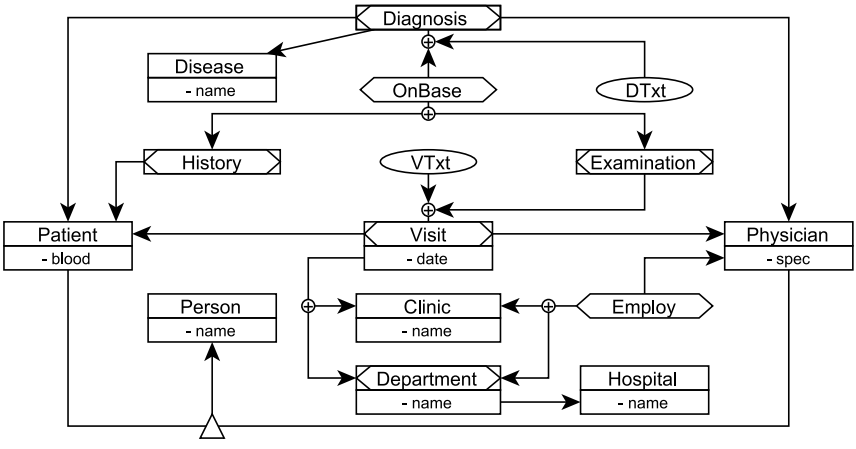

Fig. 2. XSEM-ER Schema

an element *visit* containing data about the date of the visit, the visiting patient, the visited physician, and the place of the visit. Moreover, there is a description of the visit mixed with the examinations performed during the visit.

```
<visit><date>2006-09-12</date>
  <patient><name>John Black</name></patient>
  <physician><name>Bill White</name></physician>
  <department><name>Department A</name>
    <hospital><name>Hospital B</name></hospital>
  </department>
  <description>Because of the results of a
    <examination><dsc>manual abdominal examination</dsc>...</examination>
    there is a suspicion of some liver problems. Consequently, I made a
    <examination><dsc>blood analysis</dsc>...</examination>...
  </description>
</visit>
```

Ex. 1: Mixed Content in XML

An *outgoing cluster type* represents a union of entity types and it can be used as a component of a relationship type or weak entity type. In a graphical representation, an outgoing cluster type is displayed as a circle with an inner label +. It is connected by a solid line with a relationship type or weak entity type it participates in. Each component of the cluster type is connected by an arrow going from the circle to the component.

We use outgoing cluster types for modeling irregular structure of XML. For example, patients can visit physicians at departments of hospitals and at separate

clinics. This is an example of irregular structure we can express in XML. We use an outgoing cluster type *Department*+*Clinic* to model this situation. We show the cluster type at Fig. 2. Ex. 2 is a motivating example for the introduction of outgoing cluster types. There is an element *patient* representing a patient with a name "John Black". It contains a list of *visit* elements representing patient's visits. There are two visits in the XML document. First, he visited a physician "Bill White" at a department "Department A" of a hospital "Hospital B". Then he visited a physician "Jack Brown" at a clinic "Clinic C". It is a simple example of irregular structure. There is a *department* element in the first *visit* element and a *clinic* element in the second *visit* element.

```
<patient><name>John Black</name>
  <visit><date>2006-09-12</date>
    <physician><name>Bill White</name></physician>
    <department><name>Department A</name>
      <hospital><name>Hospital B</name></hospital>
    </department></visit>
  <visit><date>2006-10-03</date>
    <physician><name>Jack Brown</name></physician>
    <clinic><name>Clinic C</name></clinic></visit>
</patient>
```

Ex. 2: Irregular structure in XML

Incoming cluster types are used for grouping different relationship types, weak entity types, and data node types having the same component. We call this component as *parent* of the incoming cluster type. The incoming cluster type specifies that instances of the components of the incoming cluster type connected with the same parent instance are mixed together. Moreover, ordering can be specified on such groups. Hence, we can use incoming cluster types for modeling mixed content in XML documents. In a graphical representation, an incoming cluster type is displayed as a circle with an inner label +. It is connected by a solid line with its parent and there is a solid arrow from each of the components to the cluster.

For example, we use an incoming cluster type $(Visit, Examination+VText)$ at Fig. 2 to model a description of a visit mixed with the examinations made during the visit. Ex. 1 described above is a motivating example. It is important to specify that the incoming cluster type is ordered because an ordering between the parts of the visit description and the examinations performed during the visit is important as shown at Ex. 1.

4.2 Hierarchical Projections

The notion of *hierarchical projections* represents the step between the non-hierarchical XSEM-ER level and the hierarchical XSEM-H level. It is a formal-

ization of binarization of relationship types and weak entity types. For example, the weak entity type $Visit$ can be represented in a hierarchy where we have a list of patients, for each patient we have the list of patient's visits, and for each patient's visit we have the visited physician and the department or clinic where the patient visited the physician. This hierarchy describes the structure of the XML document at Ex. 2.

Hierarchical projections formalize such descriptions of hierarchical organizations of non-hierarchical relationship types and weak entity types. For example, the previous hierarchy is described by the following three hierarchical projections:

$$Visit[Patient \to Visit] \quad (HP1)$$
$$Visit^{Patient}[Visit \to Physician] \quad (HP2)$$
$$Visit^{Patient}[Visit \to Department + Clinic] \quad (HP3)$$

$HP1$ represents a list of patient's visits. $HP2$ represents the visited physician and $HP3$ represent the department or clinic where the patient visited the physician. Another hierarchy is described by the following three hierarchical projections:

$$Visit[Department + Clinic \to Physician] \quad (HP4)$$
$$Visit^{Department+Clinic}[Physician \to Patient] \quad (HP5)$$
$$Visit^{Department+Clinic\,Physician}[Patient \to Visit] \quad (HP6)$$

It represents a hierarchy with a list of departments and clinics. For each department or clinic there is a list of physicians being visited by patients at the department or clinic ($HP4$). For each physician, in the context of a department or clinic, there is a list of patients who visited the physician at the department or clinic ($HP5$). Finally, for each patient in the context of a department or clinic and physician there is a list of patient's visits of the physician at the department or clinic ($HP6$).

More formally, a hierarchical projection of R is an expression $R^{context}[parent \to child]$. It specifies a hierarchy where *parent* is superior to *child*. *Context* is a sequence of components of R and specifies the context in which the projection is considered. For example, HP5 specifies a hierarchy where $Physician$ is superior to $Patient$ in the context of $Department$ or $Clinic$.

4.3 XSEM-H

XSEM-H is used for a specification of a hierarchical organization of a part of a given XSEM-ER schema using hierarchical projections. It does not add any semantics. An XSEM-H schema is an oriented graph where nodes represent entity types, relationship types, and data node types from the XSEM-ER schema and edges represent hierarchical projections of these types.

An XSEM-H schema is derived from an XSEM-ER schema by *transformation operators*. As parameters for the transformation we supply entity types, relationship types, and data node types we want to represent in the hierarchical XSEM-H schema and specify how the components should be organized in the

hierarchy. The operators are not described in a more detail here. For a more detail, see [5].

Fig. 3 shows an XSEM-H schema where the edges labeled with 1, 2, and 3 represent the hierarchical projections $HP1$, $HP2$, and $HP3$.

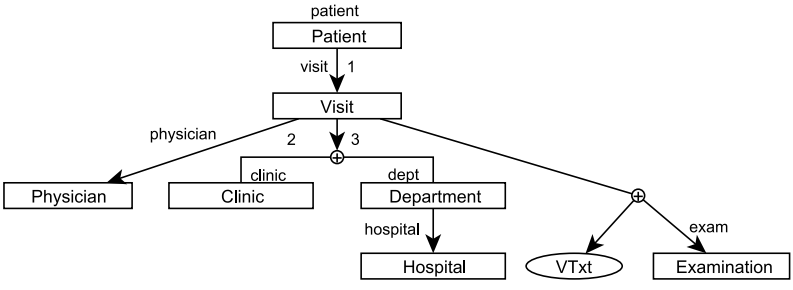

Fig. 3. XSEM-H Schema

Fig. 4 shows an XSEM-H schema where the edges labeled with 4, 5, 6, and 7 respectively, represent the hierarchical projections $HP4$, $HP4$, $HP5$, and $HP6$ respectively. At the top of the hierarchy, there is represented the outgoing cluster type $Department + Clinic$. However, we need the resulting hierarchical schema to have a tree structure. Hence, each node in the tree can have no or only one parent. For this reason we propose so called *structural representatives* in this paper as an extension to XSEM described in [5]. We represent the hierarchical projection $HP4$ by the two edges labeled with 4 and 5. Each has a child node representing *Physician*. These two child nodes have the same content which is specified by the parent of the edge labeled with 6. The child nodes of 4 and 5 are *structural representatives* of the parent of 6. The reason for this is that we specify the structure of the *Physician* representation only once and we denote the places where the *Physician* representation can be placed in the schema by one or more structural representatives.

It is important to keep a tree structure of hierarchical schemes. Otherwise, we would have problems with the schema presentation in a transparent way. Assume that the nodes representing *Department* and *Clinic* have more child nodes. Connecting the edges 4, 5 with the same child node representing *Physician* means problems with displaying the child nodes in the right order. However, this order is important when we model XML data. Moreover, such a presentation of the schema would not be so transparent in general.

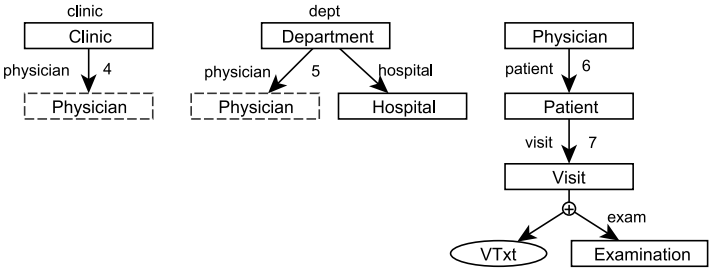

Fig. 4. XSEM-H Schema

5 Service Creation and Maintenance Support with XSEM

The goal of this paper is to show how XSEM can be applied to modeling of service XML interfaces and how it can facilitate the service creation and maintenance. We showed how we can use XSEM to model the structure and semantics of the XML data exchanged through the interfaces. We have an XSEM schema consisting of an overall non-hierarchical XSEM-ER schema describing the semantics of the data and several hierarchical XSEM-H schemes describing how the data is organized in hierarchial XML documents exchanged through the interfaces.

On the logical level we need to store the modeled data in an internal database and we need scripts for the transformation between the internal database structure and the XML structure required by the interfaces.

First, we need to derive the internal logical database schema from the XSEM schema. The process is similar to the process of derivation of relational database schemes from E-R schemes. However, this process is more complex in the case of XSEM because an XSEM schema consist of the XSEM-ER schema and one or more XSEM-H schemes. Therefore, if we want to derive an optimal database schema we have to take into account not only the non-hierarchical XSEM-ER schema but also the hierarchical XSEM-H views and optimize the logical structure of the data according to the required hierarchial organizations. There are several possibilities of how to store our data on the logical level.

(1) Native XML database systems seem as an optimal solution because we are dealing with XML data. However, if there are more XSEM-H schemes organizing the same components from the XSEM-ER schema in different hierarchies we have to choose one of them as the primary organization describing the logical schema of the native XML database and provide the scripts for the transformation between the primary organization and the other hierarchical organizations. Therefore, the performance of the system strongly depends on the ability of the native XML database system to effectively execute such transformations. However, the performance of recent native XML database systems is weaker than the performance of relational database systems.

(2) Relational database systems are very effective for strictly structured data. However, they are not too effective for semistructured data like XML. It is necessary to decompose the XML data to many tables and to join them back in different ways. It is possible to translate an XSEM-ER schema into a relational schema in a very similar way as in the case of classical E-R and to construct SQL views that build required XML documents. The advantage is that the logical database schema does not depend on required hierarchical XSEM-H views but the overall XSEM-ER schema only. Therefore, it is not a problem to add, delete, or change a particular hierarchical view. However, it can be ineffective to built frequent hierarchies repeatedly from the tables.

(3) Nor the native XML approach nor the relational approach seems to be optimal for our purposes. However, recent database systems like DB2 9 [7] offer a possibility to combine both approaches effectively. In such systems there is still a basic concept of *table* but extended with the *XML data type* supplied with an extension of SQL to build XML data from relational ones and an XQuery support. Therefore, we can combine structured and semistructured data easily in one table and combine the advantages of both approaches. Moreover, there can be many cases where data is structured but breaking the first normal form is an advantage (for attributes of entity and relationship types for example). Therefore, we can use the object-relational model instead of the relational one.

The problem to solve is what parts of the XSEM-ER schema should be represented in the object-relational model and what parts should be represented in the XML model. The basic solution is to divide the entity and relational types to two groups. The ones that are represented in more different hierarchies where it would not be effective to select one of them as the primary and transform it to the others should be represented in the object-relational model (i.e. table). The ones that are represented in only one or often repeated hierarchy where it would not be effective to construct the hierarchy frequently or the ones with a very irregular or mixed content should be represented in the XML model.

For example, the entity types *Hospital*, *Patient*, or *Physician* appear in many hierarchies with different structure. Therefore, it is better to represent them as separate tables. On the other hand, a patient history or examination have an unchanging hierarchical structure (they are documents) and their parts (not modeled in our simple XSEM-ER schema) do not appear in other hierarchies. Therefore, we can store the whole history or examination as one XML document in our database and we do not need to store their parts in separate tables.

After we have derived the logical schema from the conceptual schema we need the scripts for the transformation between the logical database structure and the XML structure of the interfaces. We have the explicit binding between the logical schema and the XSEM-ER schema, between the XSEM-ER schema and the XSEM-H schemes, and we can easily and automatically derive the XML schemes from the XSEM-H schemes. Therefore, we can automate the creation of the transformation scripts between the logical database structure and the XML interface structures.

Our approach facilitates not only the service creation but also its maintenance. Any change in the structure of the interfaces is firstly modeled in the XSEM schema. Therefore, we can propagate the change directly to the XML schemes describing the interfaces, to the logical database level, and also to the transformation scripts between them.

6 Conclusions and Future Work

In this paper, we described a new conceptual model for XML called XSEM which is based on modeling of XML schemes as views on an overall non-hierarchical schema. We described how XSEM can be used for modeling of XML interfaces of services in SOA and how it can facilitate the creation and maintenance of the services.

In our future research we will propose detailed algorithms for the translation of XSEM-H schemes to the logical XML level. Beside the grammar based XML schema languages such as XML Schema we will study the usage of pattern based XML schema languages such as Schematron [3] for the description of more complex integrity constraints. After this, we will create a prototype CASE tool for designing XSEM schemes. Moreover, we will propose algorithms for the automatic derivation of the logical database schemes from XSEM schemes and algorithms for the automatic derivation of the scripts for the transformation between the logical database structure and XML structure modeled by XSEM-H schemes as requested by this paper.

References

1. Dobbie, G., Xiaoying, W., Ling, T.W., Lee, M.L.: ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data. TR21/00, Department of Computer Science, National University of Singapore. December 2000.
2. D. C. Fallside, P. Walmsley. *XML Schema Part 0: Primer Second Edition.* World Wide Web Consortium, Recommendation REC-xmlschema-0-20041028. October 2004.
3. International Organization for Standardization, Information Technology Document Schema Definition Languages (DSDL) Part 3: Rule-based Validation Schematron. ISO/IEC 19757-3, February 2005.
4. Necasky, M.: Conceptual Modeling for XML: A Survey. Tech. Report No. 2006-3, Dep. of Software Engineering, Faculty of Mathematics and Physics, Charles University, Prague, 2006, 54 p. `http://www.necasky.net/papers/tr2006.pdf`
5. Necasky, M.: XSEM - A Conceptual Model for XML. In Proc. Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM2007), Ballarat, Australia. CRPIT, 67. Roddick, J. F. and Annika, H., Eds., ACS. 37-48., January 2007.
6. Psaila, G.: ERX: A Conceptual Model for XML Documents, *in* Proceedings of the 2000 ACM Symposium on Applied Computing, p. 898-903. Como, Italy, March 2000.
7. Saracco, C.M., Chamberlin, D., Ahuja, R.: DB2 9: pureXML Overview and Fast Start, IBM Redbooks, 134 p., June 2006.
8. Sengupta, A., Mohan, S., Doshi, R.: XER - Extensible Entity Relationship Modeling, *in* Proceedings of the XML 2003 Conference, p. 140-154. Philadelphia, USA, December 2003.