Transformation of DEMO Metamodel into XML Schema

Yan Wang[1], Antonia Albani[2], and Joseph Barjis[3]

[1] Tilburg University
European Research Institute in Service Science
PO Box 90153, 5000 LE Tilburg, The Netherlands
y.wang13@uvt.nl
[2] University of St. Gallen
Institute of Information Management
Müller-Friedberg-Strasse 8, 9000 St. Gallen, Switzerland
antonia.albani@unisg.ch
[3] Delft University of Technology
Faculty of Technology, Policy and Management
PO Box 5031, 2600 GA Delft, The Netherlands
j.barjis@tudelft.nl

Abstract. In this paper, we propose an approach to transform models derived by applying the Design and Engineering Methodology for Organizations (DEMO) into an exchangeable format. DEMO is based on a founded theory, the Ψ-theory, and satisfies the requirements to be a well defined domain modeling methodology. Having the DEMO models represented in an exchangeable format is beneficial for different types of applications supporting the information system development process. Applications used for the automatic analysis (simulation) of the DEMO models or for the identification of business components are just two examples to be mentioned.

Keywords: DEMO metamodel, model transformation, XML Schema.

1 Introduction

Modern enterprises are challenged by the reality of a dynamic business environment. With the increasing business scale, companies are encountering more complex situations, such as globalized sales and sourcing markets, shortened product life cycles, and innovative pressure on products, services and processes [1]. In order to stay in and win the game in the competitive business world, enterprises need to adapt to the fast changing market quickly and expand their cooperative relationships with their business partners. Adaptive and agile information systems for enterprises are therefore crucial in order to support the business needs. Thus while developing such information systems for enterprises, it is necessary to have a suitable methodology for modeling the business domain. The appropriateness and the quality of the business domain models are vital for the

A. Albani, J.L.G. Dietz, and J. Verelst (Eds.): EEWC 2011, LNBIP 79, pp. 46–60, 2011.

development process. Dietz proposes some quality criteria regarding a business domain model in [2], which are *coherence* (the domain models constitute a logical and truly integral whole), *comprehensiveness* (complete coverage of all relevant issues), *consistency* (the domain models are free of contradictions or irregularities), *conciseness* (all relevant models are compact and succinct), and *essence* (the domain model should only show the essence of the enterprise). The *Design and Engineering Methodology for Organizations (DEMO)* [2] is a methodology that satisfies these requirements and that additionally distinguishes between essential (business), infological and datalogical acts and facts, ultimately resulting in the distinction between three aspect organizations: the B-organization (B from business), the I-organization (I from infological), and the D-organization (D from datalogical). The apparent advantages of DEMO – in particular the huge reduction of complexity [3] – led us to choose DEMO for modeling the business domain.

DEMO has already proven to be beneficial for the development of the supporting information systems (see [4, 5]). However no exchange format of DEMO models has been defined so far in order to allow for automatic access of DEMO model data by third party applications. Example applications are a) tools for the automatic identification of business components based on DEMO models, or b) simulation tools, where readability and structure of the model data are both important and helpful. This paper therefore presents research on metamodel transformation, where the transformation of DEMO metamodel to XML Schema is elaborated.

The paper is structured as follows. Section 2 introduces state of the art in model transformation. Additionally, the section presents the metamodel transformation approach as used in this paper. The DEMO metamodel which is needed for the said transformation is introduced in the sections 3. In section 4 several transformation rules that are used to transform the DEMO metamodel into XML Schema are introduced. One exemplary case, in section 5, shows the transformation procedure and the resulting XML structure for an instance of a DEMO metamodel. The evaluation of the transformation results is made in section 6. At last, section 7 concludes and proposes some work for future research.

2 Model Transformation

Metamodeling has been widely used as a transformation approach, of which exemplary applications can be found in [6] [7]. Figure 1 visualizes a general metamodel mapping transformation. The transformation is from Model A to Model B. The language used for expressing Model A is defined in Metamodel A. The Metamodel B defines the language that describes Model B. These two languages are connected by the Transformation model. The task of this transformation model is to map the source language and the target language, so that it can build a bridge from Model A to Model B. The transformation model is expressed in terms of a set of transformation rules that address the detailed guidance of the transformation. Thus the transformation procedure from Model A to Model B is accomplished by using metamodeling.

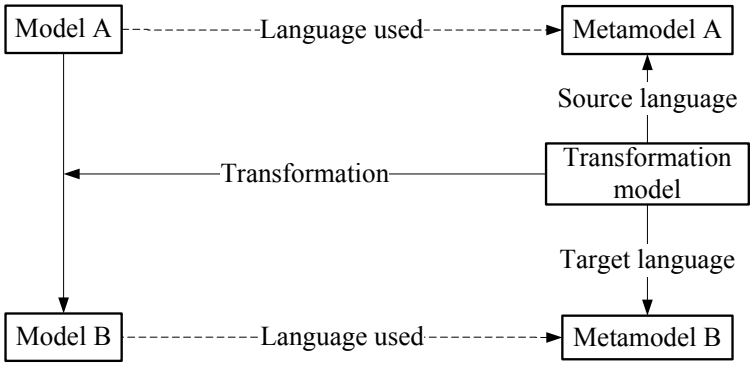

Fig. 1. Metamodel Mapping Transformation [8]

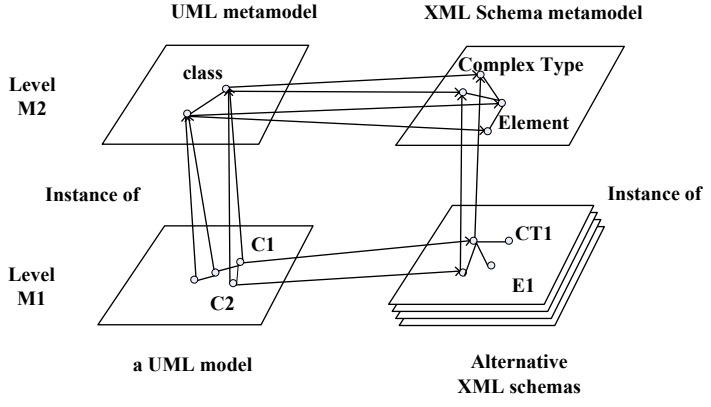

Fig. 2. UML to XML Schema Transformation [9]

A good example of applying metamodeling e.g., in the Model Driven Architecture (MDA), is the transformation between UML class diagram and XML Schema from [9]. Depicted in Figure 2, the source UML model and the target alternative XML Schema are situated in level M1, the source UML metamodel and the target XML Schema metamodel are situated in level M2. The source and target models are mapped with each other within the same level. For example, the class construct from UML metamodel, which is in level M2, is possible to be mapped either to "Element" declaration or "Complex Type" definition construct from XML Schema metamodel, which is also in level M2. Following the same transformation rules, instances as e.g., "C1" or "C2" of the UML metamodel "class" type can then be mapped either to instances of the "Element" or "Complex Type" of the XML Schema metamodel.

When using the metamodeling approach in model transformation, the actual transformation is executed within the modeling languages used in the metamodel

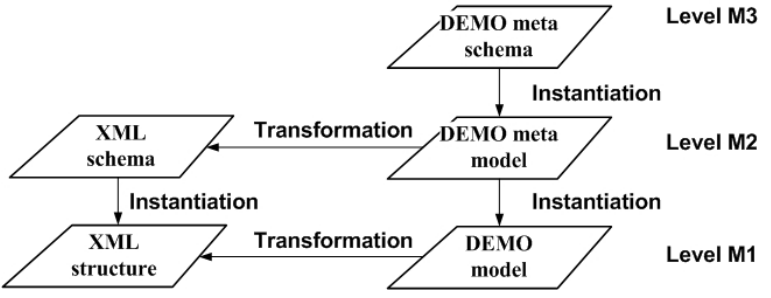

Fig. 3. DEMO metamodel to XML Schema transformation

of the source and target models. The complex instance contents at the model level are left out of sight during the transformation. In the presented research we use the DEMO metamodel defined by Dietz and introduced in [10] as the source, and the XML Schema as the target metamodel on the level M2. Fig. 3 illustrates the elements used for the transformation needed. The DEMO meta schema on the level M3 defines the concepts used for defining the DEMO metamodel. Based on the DEMO metamodel and the XML Schema on level M2, transformation rules are defined allowing the transformation of DEMO models into XML format on level M1.

3 DEMO Metamodel

As metamodeling transformation approach is chosen in this proposed DEMO transformation, DEMO metamodel is a crucial source material for the transformation. The DEMO metamodel [10] on level M2 (see Fig. 3), corresponding to the aspect models on level M1, includes the Meta Construction Model (MCM), Meta Process Model (MPM), Meta Action Model (MAM), Meta State Model (MSM), and the metamodel for cross-model tables TRT, IUT and BCT. These metamodels provide the basic constructs and relationships that occur in any instance aspect of DEMO models on level M1 (see fig. 3). The language used in specifying the DEMO metamodel is called World Ontology Specification Language (WOSL). Regarding the limited space in this paper, we restrict ourselves on describing two aspects of the DEMO metamodel, namely the MSM and the MCM. For the same reason, only these two are discussed in the following sections. We refer to [10] for readers who are interested in a complete description of DEMO metamodel.

3.1 Meta State Model (MSM)

We start by introducing the MSM, since it builds the basis for the whole DEMO metamodel. The MSM is part of the DEMO metamodel, and has connection with

the MCM. It fulfills two roles, besides being the metamodel of SM, MSM also defines the metamodel of the whole DEMO metamodel. We call the metamodel of the whole DEMO metamodel a *meta schema* (see level M3 in Fig. 3). We are going to briefly describe the MSM's role of being the meta schema and focuse on its specification of being metamodel of SM, because the said transformation takes place on level M2 (Fig. 3).

The MSM, as DEMO meta schema, specifies the concepts used in WOSL, including the declared fact type, derived fact type, unary fact type, binary fact type, object class, scale and category. Furthermore, existence laws, such as exclusion law, unicity law, reference law and dependency law are defined in this meta schema as well.

The MSM, as being the metamodel of SM, defines the basic constructs of the SM. A SM is instantiated in terms of a lawful set of basic constructs. Figure 4 shows how the MSM specifies a basic construct, and how a basic construct in SM look like. The basic construct in SM (Figure 4 (b)) states that the role x in fact type F has a domain A. And for every a ∈ A there must be a tuple <a>in F. The corresponding definition of this construct in MSM is expressed in Figure 4 (a): for every unary fact type there must be one and only one instance fact type x in the binary fact type <x, y>, the domain of fact type x is the object class y. Additionally, object x is dependent on object y, while x belongs to class UNARY FACT TYPE, and y belongs to an extension of unary fact type.

The two roles of MSM imply that the MSM not only defines the structure of instance SM, but also defines the concepts used by the whole DEMO metamodel. The MCM in the following section will be explained by using the concepts defined in the MSM.

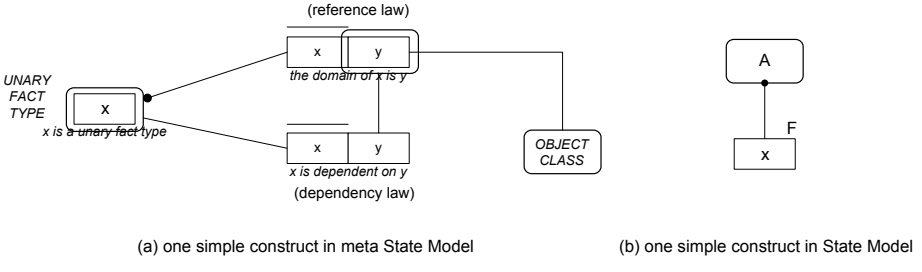

Fig. 4. Basic construct at both metamodel level and model level (with dependency law)

3.2 Meta Construction Model (MCM)

The MCM specifies the *interaction structure* of CM. Being outlined in Figure 5, TRANSACTION KIND, ELEMENTARY ACTOR ROLE and INFORMATION BANK including the PRODUCTION BANK and COORDINATION BANK are the core object classes within the MCM.

For describing the DEMO metamodel we adopt the following convention (we take the type ELEMENTARY ACTOR ROLE as an example): if there is a fact

a, and the type of a is the elementary actor role, we would say that 'a is an elementary actor role' or 'elementary actor role a'. This is equivalent saying that 'elementary actor role (a) holds'. This kind of expression will be used in the specification of DEMO metamodel in this section.

The description of the correlation between the ELEMENTARY ACTOR ROLE and TRANSACTION KIND is specified like this: if there is an 'elementary actor role a is an initiator of transaction kind t', there must be a 'transaction kind (t) holds'. Meanwhile, constrained by the dependency law, for every transaction kind, there must be an elementary actor role a, that 'a is an initiator of t holds', where t is the transaction kind. If there is an elementary actor role 'a is the executor of transaction kind t', there must be a 'transaction kind (t) holds'. Meanwhile, constrained by the dependency law, for every transaction kind, there must be an elementary actor role a, that 'a is the executor of t holds', where t is the transaction kind. Note that there are unicity laws hold for both a and t in the lawful binary fact type. That means every transaction kind and every elementary actor role cannot occur more than once in the lawful binary fact type. Thus it implies that there is a strict one-to-one relationship between the transaction and its executor.

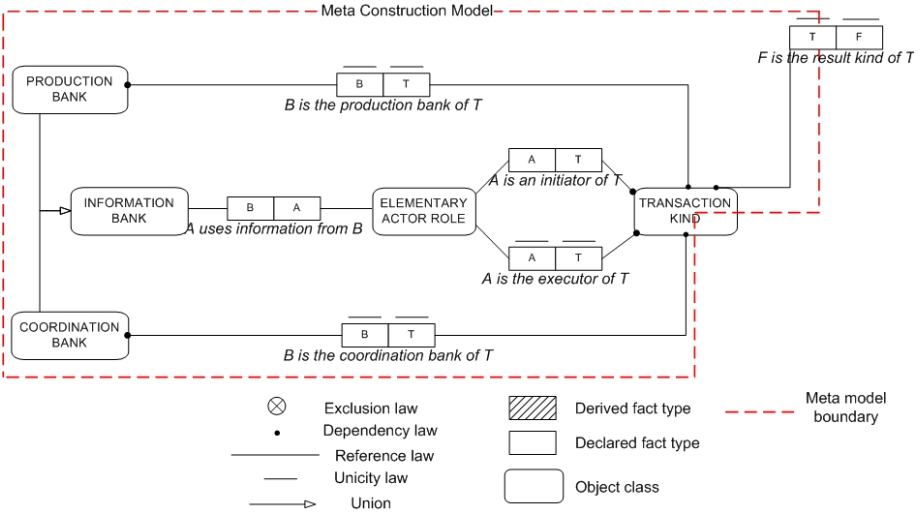

Fig. 5. Meta Construction Model

The MCM also specifies that an elementary actor role, either initiator or executor, may use information from information banks, but not necessarily. There are two kinds of information banks, namely, the production bank and coordination bank. The correlation between transaction kinds and information banks are restricted to a one-to-one relationship, as shown in Fig. 5 there are unicity laws hold in the binary fact types between transaction kinds and information banks respectively. In addition, constrained by the dependency laws hold for transaction

kinds and information banks, every coordination fact or production fact produced by a transaction must belong to the corresponding coordination bank or production bank respectively; every coordination bank or production bank can only come into existence when there is some coordination fact or production fact respectively. It is not possible to have an empty information bank or a transaction without its corresponding information banks, since there are always coordination facts and production facts produced as the results of the coordination acts and production acts which are performed during transaction process steps.

The result types are the connections between the MCM and the MSM. The transaction kind plays a crucial role in connecting MCM to MSM. Seen in Fig. 5, in the binary fact type 'F is the result kind of T', transaction kind T belongs to TRANSACTION KIND which is part of MCM. Result kind F, situates outside the MCM boundary, belongs to declared fact type which is part of MSM. The connection between the transaction kind and its result kind is strictly one-to-one relationship, as marked by the unicity laws.

An exemplary instance CM of a library case include an Organization Construction Diagram (OCD) (Figure 6) and a TRT (Table 1). In the exemplary transaction 'membership registration' (T01), the initiator of T01 is the composite actor role 'aspirant member' (CA02); the executor of T01 is the elementary actor role 'registrar' (A01). The composite actor role could be an elementary actor role or a composite actor role. The transaction symbol has two meanings in OCD, one is the transaction T01, the other one is the combination of production bank PB01 and coordination bank CB01 that belong to transaction T01. The elementary actor role A01 uses information from the composite production bank 'personal data' (CPB11) and 'general data' (CPB14). For T01, the result type is R01 'membership M has been started'. This structure is specified in MCM,which implies that the relationship between actor roles and transaction in the instance model is fully consistent with the corresponding structure in the metamodel.

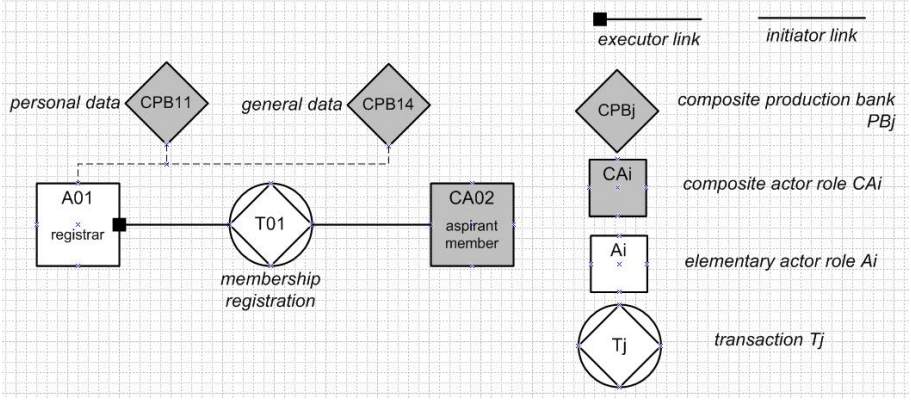

Fig. 6. An exemplary OCD of the library

Table 1. The TRT of the library

transaction type	**result type**
T01 membership registration	R01 membership M has been started

4 Transformation Rules

The transformation rules will be derived from three aspects: what information needs to be transformed (selection rules); how the selected information is structured in target format (structuring rules); how to verify the transformation results (mapping rules).

4.1 Selection Rules

Selection rules aim to clarify the complete and essential information objects for the transformation.

In this paper, we select information from MCM and MSM. For MCM, we select the object classes, binary fact types and the correlation among them. It is because the object classes contain all the information objects, and the binary fact types and the correlations contain the relationships among the information objects.

Considering the fact that the proposed transformation is about DEMO metamodel on level M2 (Fig. 3), we only pick up the object classes and fact types that are used in the basic constructs, and the existence laws which are directly used in building the DEMO metamodel, but exclude the concepts for specifying object class, fact types and existence laws are situated on level M3, namely DEMO meta schema. Tables 2 lists all the information objects chosen from MCM and MSM. The selected information will be transformed into target XML-based format.

Table 2. The selected information from the MCM and MSM

	Information Items		**Information Items**
MCM	TRANSACTION KIND	**MSM**	FACT TYPE
	A is an initiator of T		OBJECT CLASS
	A is the executor of T		the domain of x is y
	ELEMENTARY ACTOR ROLE		x is a declared fact type
	A uses information from B		SCALE
	INFORMATION BANK		PROPERTY
	PRODUCTION BANK		c is the domain of x
	B is the production bank of T		s is the range of x
	COORDINATION BANK		mutual exclusion holds for x and y
	B is the coordination bank of T		x is dependent on y
	F is the result kind of T		unicity holds for x

4.2 Structuring Rules

The chosen contents from the DEMO metamodel must be structured hierarchically in the XML schema files. It implies that those information objects must be clustered as elements, complex types or attributes in XML Schema, regarding their different features and priorities in DEMO metamodel. An example of the general structure of XML document is shown in Listing 1.1.

Listing 1.1. An example of defining the element and complex type and attribute

```
<xsd:element name="ELEMENT" type="COMPLEX_TYPE"/>
<xsd:complexType name="COMPLEX_TYPE">
    ...
<xsd:attribute name="attribute" type="type" use="required/optional"/>
</xsd:complexType>
```

We define those which have the central position in the metamodel as the root element in the schema, e.g. the one which holds the most dependency laws, or the one which is the most generically constructed in the metamodel. For instance, the transaction kind has the central position in MCM, which is because in MCM the transaction kind holds the most dependency laws. Its central position also makes sense at the model level, since in DEMO aspect models, CM is the most concise model as mentioned in section 3.2, while the other models detail part of the CM. The other information, which is not in the central position, is defined as complex types in order to detail the root element. The existential constraints contained in the metamodels are defined as attributes of either elements or complex types.

Besides considering the hierarchy in the chosen contents, we also pay attention to the structure of the instance XML files. We need to add an additional root element to have the structured model information nested in the root element, when the current root element is instantiated by more than one cases.

Listing 1.2. XML schema for global element <Transactions>

```
<xsd:element name="Transactions">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="Transaction" type="TRANSACTION_KIND" maxOccurs="
       unbounded"/>
   <xsd:element name="Transaction" type="TRANSACTION_KIND" maxOccurs="
       unbounded"/>
   <xsd:element name="Transaction" type="TRANSACTION_KIND" maxOccurs="
       unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

An example of the root element in the schema of the CM explains itself (Listing 1.2). The child element <Transaction>and its complex type content "TRANSACTION_KIND" comprehends all the model information within the transaction. If there are three transactions T01, T02 and T03, for each transaction, it is specified as an element <Transaction>and includes its complext type "TRANSACTION_KIND". Then we need to set another parent element for this sequence of <Transaction>. Thus we set element <Transactions>as the root element, which includes all the <Transaction>elements in the instance XML

file. This structuring rules are applied to transform the model information from source format to target format.

4.3 Mapping Rules

This rule is made to guarantee the information completeness and correctness in the transformation results, in order to verify the transformation results. The mapping is made from two perspectives, which are the precision of the constraints in the XML schema files and the completeness of the information objects in the instance XML documents, compared with the original DEMO metamodel. Therefore, for each information object in the original DEMO metamodel, there should be a corresponding interpretation in the XML schema files, in terms of either an element or a complex type; for every necessary constraint, there should also be an equivalent part in the schema files, in terms of an attribute of either an element or a complex type.

The following table lists how the proposed transformation converts DEMO meta schema elements in XML Schema elements at a high level. The DEMO meta schema elements are the elements specified in WOSL, including the object class, the fact type, the exclusion law, the unicity law, the reference law and the dependency law. The object class is converted to either an element or a complex type content. The fact type is considered as either a child element or a contained element within complex types. The exclusion law restricts the value of an XSD string simple type. The unicity law notifies the occurrence of a contained element within complex types. The reference law and the dependency law tell the necessity of the usage of an attribute.

Table 3. DEMO meta schema elements and XML Schema elements

DEMO meta schema elements	**XML Schema elements**
Object class	Element Complex type
Fact type	Child element of a contained element (complex type)
Exclusion law	Restriction of an XSD string simple type with two enumeration facets
Unicity law	Occurrence indicators of elements •minOccurs = 1 •maxOccurs = 1
Reference law	Attribute of an element •use = optional
Dependency law	Attribute of an element •use = required

5 Exemplary Case

In this section, we take one transaction of the library case as an example to demonstrate the transformation procedure. As exemplified in Figure 6 and Table

1 (section 3), CA02 (aspirant member) initiates T01 (membership registration), of which the executor is numbered A01 (register). The result type of T01 is that R01 membership M has been started. The coordination and production bank that belong to T01 are combined in the information bank PB01. In addition, internal actor role A01 gets to know some general information from CPB14, and the personal information about the aspirant member from CPB11.

Listing 1.3. Schema code for Meta Construction Model

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="Transactions">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="Transaction" type="TRANSACTION_KIND" maxOccurs="
       unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
<xsd:complexType name="TRANSACTION_KIND">
 <xsd:sequence>
  <xsd:element name="Tname" type="xsd:string"/>
  <xsd:element name="Initiator" type="ELEMENTARY_ACTOR_ROLE" minOccurs=
      "1" maxOccurs="unbounded"/>
  <xsd:element name="Executor" type="ELEMENTARY_ACTOR_ROLE" minOccurs="
      1" maxOccurs="1"/>
  <xsd:element name="UseInformation" type="INFORMATION_BANK" minOccurs=
      "1" maxOccurs="1"/>
  <xsd:element name="Result" type="DeclaredFactType" minOccurs="1"
      maxOccurs="1"/>
 </xsd:sequence>
 <xsd:attribute name="TransactionID" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="ELEMENTARY_ACTOR_ROLE">
 <xsd:sequence>
  <xsd:element name="name" type="xsd:string"/>
  <xsd:element name="UseInformation" type="INFORMATION_BANK" minOccurs=
      "0" maxOccurs="unbounded"/>
 </xsd:sequence>
 <xsd:attribute name="ActorID" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:complexType name="INFORMATION_BANK">
 <xsd:simpleContent>
  <xsd:extension base="xsd:string">
   <xsd:attribute name="BankType" use="optional">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Production"/>
      <xsd:enumeration value="Coordination"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
   <xsd:attribute name="BankID" type="xsd:string" use="required"/>
  </xsd:extension>
 </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="DeclaredFactType">
 <xsd:simpleContent>
  <xsd:extension base="xsd:string">
   <xsd:attribute name="ResultID" type="xsd:string" use="required"/>
  </xsd:extension>
 </xsd:simpleContent>
</xsd:complexType>
</xsd:schema>
```

The MCM is the corresponding metamodel for the instance model described above. Applying the selection rules, we have the information objects that need to be transformed (Table 2). With the guidance of the structure rules, we design the XML Schema (Listing 1.3) as the transformation result of those selected information objects from the MCM.

In order to elaborate the transformation as automatically as possible, an application, which can produce the XML files for DEMO models based on the designed XML schema and input model information, is built. A form (Figure 7) is designed to collect the required information objects from instance CM (Figure 6,Table 1). All the filled information will be processed in the application and structured in a new XML file (Listing 1.4), of which the structure is defined in the above schema (Listing 1.3).

Listing 1.4. The XML document of CM

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Transactions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="C:\Tomcat\webapps\ROOT\demo\CM.xsd">
<Transaction TransactionID="T01">
 <Tname>membership registration</Tname>
 <Initiator ActorID="CA02">
  <name>aspirant member</name>
   <UseInformation BankID="PB01">PB01</UseInformation>
  </Initiator>
  <Executor ActorID="A01">
   <name>registrar</name>
   <UseInformation BankID="PB01">PB01</UseInformation>
   <UseInformation BankID="CPB14" BankType="Production">general data</
       UseInformation>
   <UseInformation BankID="CPB11" BankType="Production">personal data</
       UseInformation>
  </Executor>
  <UseInformation BankID="PB01">PB01</UseInformation>
  <Result ResultID="R01">membership M has been started</Result>
</Transaction>
```

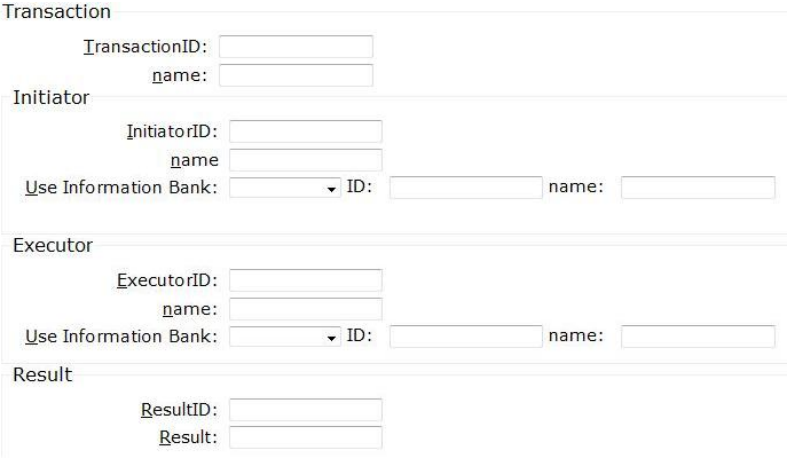

Fig. 7. The information items required for CM

This is one simple example to demonstrate the transformation procedure and show the result of our proposed DEMO transformation. In next section, an evaluation will be made to this transformation result.

6 Evaluation

In this section, we will evaluate the proposed transformation from two perspectives, which are the verification and the usefulness of the transformation result respectively.

6.1 Verification

Verifying the transformation results is of significant importance to the feasibility of the proposed transformation approach. We will present a comparison of the produced exemplary XML documents and their corresponding original DEMO models. The comparison aims to map the information contained in the XML documents with the information from the original models, to see whether the information is completely and precisely maintained during the transformation.

The comparison between the XML document and the original diagram of CM is carried out in the unit of business transaction, including the actor roles that participate in the transaction and the information banks belonging to or used by particular transaction (Figure 6). In section 5 we provide the semantic description and the XML document of this transaction. Here we map them precisely in Table 4.

Clearly seen from the comparison, all the items in the original CM have been correspondingly stored in the XML document, which implies that the

Table 4. Mapping of T01 membership registration

Description of original CM	XML document
T01 membership registration	<Transaction TransactionID="T01"> <Tname>membership registration</Tname>
CA02 (aspirant member) initiates T01 information bank PB01 belong to transaction T01 	<Initiator ActorID="CA02"> <name>aspirant member</name> <UseInformation BankID="PB01">PB01</UseInformation> </Initiator>
the executor is A01 (register)	<Executor ActorID="A01"> <name>registrar</name>
information bank PB01 belong to transaction T01	<UseInformation BankID="PB01">PB01</UseInformation>
A01 gets general information from CPB14	<UseInformation BankID="CPB14" BankType="Production">general data</UseInformation>
A01 gets personal information from CPB11	<UseInformation BankID="CPB11" BankType="Production">personal data</UseInformation> </Executor>
information bank PB01 belong to transaction T01	<UseInformation BankID="PB01">PB01</UseInformation>
result type of T01 is R01	<Result ResultID="R01">membership M has been started</Result> </Transaction>

model transformation is completely and correctly maintained during our proposed transformation procedure.

6.2 Usefulness

The XML documents of the DEMO aspect models resulting from the transformation process as introduced in the previous sections, should be of use for several types of applications. As already mentioned in the introduction, the resulting XML documents have added value for simulation tools as well as for tools used to identify business components allowing for information systems to be modeled at a high-level of abstraction. It goes beyond the scope of this paper to show how the transformation results are used for such kind of applications. We refer to [10] for further details on the usage of the transformation results for the identification of business components.

7 Conclusion

This paper discusses an approach to transform DEMO metamodels into XML Schema, allowing to export the semantic data of DEMO models into XML documents. This data can then be extracted by several different applications for further processing.

The approach involves a research about background information on model transformation, a grasp of DEMO metamodel, the definition of transformation rules, the design of XML schema for the DEMO metamodel, the instantiation of the designed XML schema, and the transformation results evaluation. A comprehensive interpretation of the DEMO metamodel was made in XML Schema. We made three transformation rules for the design of XML schema, which specify the content of the proposed DEMO transformation, determine the structure of the expected XML documents, and guarantee the completeness of the selected information and the preciseness of the transformed information. The designed XML schema can be instantiated into a set of XML documents with concrete model information as e.g., from the exemplary case. The produced XML documents are the transformation results. By semantically mapping the information items in the XML documents with the ones in the original aspect models, we verify the transformation results and guarantee the information completeness during the transformation procedure.

Based on this paper, some future works are envisioned. Firstly, reproducing the DEMO diagrams from the transformation results is expected, which provides a more direct verification of the DEMO transformation. Secondly, an automation for combining e.g., graphical user interfaces, used for producing DEMO models, and the tool, used to identify business components, is desired to be investigated in future research. The integration of these tools, based on the XML transformation of DEMO models as presented in this paper, will support designers in generating high-level constructional information system models, by means of business components. Thirdly, the simulation of DEMO models is another direction into which we want to continue our research. Any conceptual

model developed for simulation, will impact all aspects of the subsequent simulation study such as the simulation model development speed, the validity of the model, the experimentation and the confidence based in the model and future reusability of the models. Having an input conceptual model based on theory, as the Ψ-theory DEMO is based on, and having the models transformed into an exchangeable format (XML), lays down a profound opportunity for generating simulation models that can be automatically analyzed. Furthermore the XML format is simulation environment independent, which allows the simulation models to be generated using any simulation environments or tools. Although the simulation is not discussed in details in this paper, it opens up a potential future research, which will be investigated by the authors.

Acknowledgments

This project is supported by the Swiss National Science Foundation (SNSF).

References

1. Hamel, G., Prahalad, C.K.: The Core Competence of the Corporation. In: Strategische Unternehmungsplanung Strategische Unternehmungsführung, Springer, Heidelberg (2006)
2. Dietz, J.L.: Enterprise Ontology Theory and Methodology. Springer, Heidelberg (2006)
3. Dietz, J.L.: The deep structure of business processes. Communications of the ACM 49(5) (May 2006)
4. Albani, A., Dietz, J.L.: Enterprise ontology based development of information systems. International Journal of Internet and Enterprise Management, Special Issue on Enterprise Design and Engineering 7(1) (2011)
5. Albani, A., Dietz, J.L.: The benefit of enterprise ontology in identifying business components. In: The Past and Future of Information Systems: 1976–2006 and Beyond, IFIP 19th World Computer Congress, TC-8, Information System Stream, Santiago de Chile, Chile. IFIP International Federation for Information Processing, vol. 214, pp. 243–254. Springer, Boston (2006)
6. Goknil, A., Topaloglu, Y.: Ontological perspective in metamodeling for model transformations. In: MIS 2005: Proceedings of the 2005 Symposia on Metainformatics, ACM, New York (2005)
7. Koch, N.: Transformation techniques in the model-driven development process of uwe. In: ICWE 2006: Workshop Proceedings of the Sixth International Conference on Web Engineering, ACM, New York (2006)
8. OMG, O.M.G.: Mda guide version 1.0.1 (June 2003)
9. Kurtev, I., van den Berg, K., Aksit, M.: Uml to xml-schema transformation: a case study in managing alternative model transformations in mda. In: Forum on specification and Design Languages (FDL 2003), ECSI, Frankfurt, Germany (2003)
10. Wang, Y.: Transformation of demo models into exchangeable format. Master's thesis, Delft University of Technology (April 2009)