Advances in Intelligent Systems Research, volume 153

International Conference on Applied Mathematics, Modeling and Simulation (AMMS 2017)

Unified Hierarchical Model of the Data Exchange

Tatiana Penkova[1,2,*] and Sergey Kochetkov[1]

[1]Institute of Computational modelling SB RAS, Krasnoyarsk, Russia

[2]Siberian Federal University, Krasnoyarsk, Russia

[*]Corresponding author

***Abstract*—The paper presents an approach to standardized data exchange between heterogeneous resources based on a unified hierarchical model of the data representation. The paper gives a metamodel and description of the structural elements of the unified model. Some conversion algorithms were developed to turn the information storage schemes into the unified hierarchical model in the case of XML-format.**

Keywords-data exchange; unified hierarchical model; conversion algorithms; XML

I. INTRODUCTION

Compatibility of corporate systems is one of the essential conditions of business process automation. Complex universal automation systems now tend to be replaced by ad-hoc customized software products aimed at tackling specific tasks [1-4]. Nevertheless, such systems must be able to interact with one another and exchange data with external resources. Therefore it becomes necessary to develop techniques and tools for cross-system integration of information.

Informational cross-system interaction can be represented by the set of interacting subsystems and the set of informational objects taking part in the data exchange [5]. Data exchange is takes place between two interacting subsystems, one of which plays a role of a sender and another one plays that of a receiver. Each of the subsystems uses one of the formats of data exchange (XML, Relational Data Base, DOCX). The task of information integration is to provide data exchange regardless of the format used by interacting subsystems. The structure of data representation is defined by the subsystem either sending or receiving information, and the object of informational exchange. As a rule, data exchange problem is tackled by means of some extra programming moduls that convert data from one format to another directly for each object of data exchange [6, 7, 8]. Using this approach makes it necessary to create new moduls (if new objects or subsystems have been added to the data interaction system) and reprogramming (if data storage schemes have been changed). In this case the unified representation of information exchange structures of various formats makes it possible to automatise the matching of information storage schemes of items in different subsystems.

This paper presents a novel approach to standardised data exchange between heterogeneous resources based on a unified hierarchical model of the data representation. The paper gives a metamodel and description of the structural elements of the unified model. Some conversion algorithms were developed to turn the information storage schemes into the unified hierarchical model in the case of XML-format.

II. UNIFIED HIERARCHICAL MODEL OF THE DATA REPRESENTATION

The unified model presents the structure of the information object (data packet) as a hierarchy of structural elements. Figure 1 illustrates the metamodel of a unified representation of data exchange structures in the form of a class diagram in UML (Unified Modeling Language) that describes the main entities and the relationships between them. The metamodel specifies the classes of structural elements of the unified model that independent of the implementation way including the format and data storage scheme.

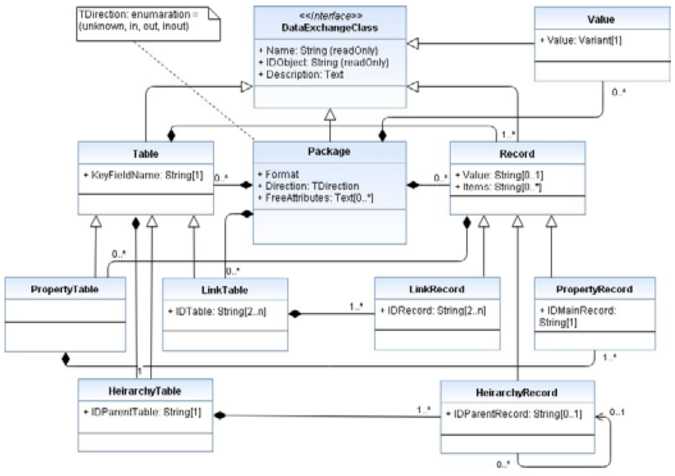


FIGURE I. METAMODEL OF THE UNIFIED MODEL

The relationship between entities is specified by the corresponding type of relationship. The relation of generalization takes place between entities in the case when one entity (subtype) is a part form of another entity (supertype). Generalizations in the model form a strict partial order. Graphically, this relation is represented by a line with an empty triangle on the side of the entity-supertype. The relation of composition defines a rigid dependence of the existence time of the entity-part on the entity-aggregate. Graphically, this relation is represented by a line with a shaded diamond on the part of the entity-aggregate. The relation of association reflects the existence of the relationship between entities and it is represented by a line with a number of entity's instances. Let us consider the main classes, their attributes and the nature of the relations between the elements of the metamodel.

0. "DataExchangeClass" is a basic class of any element included in the data packet. The required attributes of the class are: "Name" that specifies the name of the object; "IDObject" is

a unique identifier that allows to uniquely identify the object in the information exchange process; "Description" that contains information about the composition and purpose of the data.

1."Package" is a class of the root element of the object structure. The element contains information about the technical characteristics of the data packet. Required attributes of the class are: "Format" is a description of the format of the object (XML, DOCX, RDB, etc.); "Direction" is the direction of data transfer (input data, output data, round-way exchange). Additionally, the class can contain some additional attributes. The nested elements of the class "Package" are classes: "Value", "Record" and "Table". At the same time, at least one of these classes must be present.

2. "Value" is a "single value" element that can contain the values of any type including text, picture, LOB, etc.

3. "Record" is a "record" element that can contain a value accompanied by a set of attributes (fields). A collection of "Record" elements with the same set of attributes can be included in the table (class "Table"). A single "Record" element can be presented separately. The "Record" element can include one or more property tables (the "PropertyTable" class) that contain additional information about the element. The nested elements of the "Record" class are the classes: "PropertyRecord", "LinkRecord" and "HierarchyRecord".

3.1 "PropertyRecord" is an element that represents one of the kinds of "record" element and contains additional information about the parent record. The element is nested in the "PropertyTable" class. The required attribute of the class is "IDMainRecord" that presents the unique identifier of the parent record.

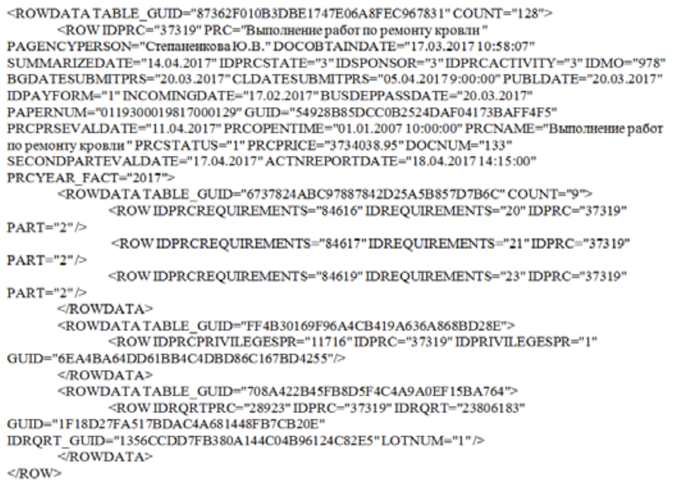
4. "Table" is the "table" element that contains one or more nested "Record" elements. Nested elements must have the same set of attributes. The table must have an attribute (field) to identify the records (key field).

4.1. "PropertyTable" is an element that represents a variation of the "Table" element and contains one or more records with additional information about the parent record (records) in form of "PropertyRecord" element.

4.2. "LinkTable" is an element that represents a variation of the "Table" element and identifies links between records of two or more tables. The required attribute of the class is "IDTable" that presents the set of unique identifiers for the linked tables. An element contains one or more "LinkRecord" elements.

4.3. "HierarchyTable" is a "hierarchy" element that represents a variation of the "Table" element and forms a hierarchy of records. The required attributes of the class are: "IDParentTable" is the unique identifier of the table for which the hierarchy of records is formed. An element contains one or more "HierarchyRecord" elements.

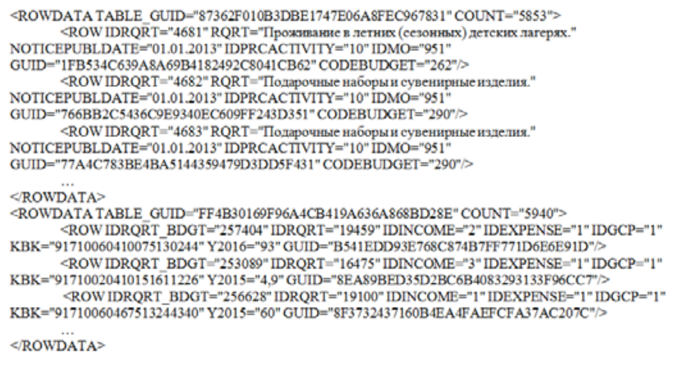
There are some ways to describe the records in XML-file as an information object. Let us consider some examples of the data representation in XML format for one of the tasks where data exchange between heterogeneous resources are especially asked – submission of bids for government and municipal procurement [9, 10]. Figure 2 shows the fragment of XML-file where the table record is specified in form of several records from property tables (example 1). Identification of objects of information exchange is performed using GUID attributes that uniquely determine the tables and the records in tables. For instance, the *ROWDATA TABLE_GUID = "87362F010B3DBE1747E06A8FEC967831"* uniquely identifies the table in different systems in the case when the table has different names. *GUID = "6EA4BA64DD61BB4C4DBD86C167BD4255"* allows to uniquely identify a data record in the case when the key field has changed while data processing.



```
<ROWDATA TABLE_GUID="87362F010B3DBE1747E06A8FEC967831" COUNT="128">
        <ROW IDPRC="37319" PRC="Выполнение работ по ремонту кровли "
PAGENCYPERSON="Степаненкова Ю.В." DOCOBTAINDATE="17.03.2017 10:58:07"
SUMMARIZEDATE="14.04.2017" IDPRCSTATE="3" IDSPONSOR="3" IDPRCACTIVITY="3" IDMO="978"
BGDATESUBMITPRS="20.03.2017" CLDATESUBMITPRS="05.04.2017 9:00:00" PUBLDATE="20.03.2017"
IDPAYFORM="1" INCOMINGDATE="17.02.2017" BUSDEPPASSDATE="20.03.2017"
PAPERNUM="0119300019817000129" GUID="54928B85DCC0B2524DAF04173BAFF4F5"
PRCPRSEVALDATE="11.04.2017" PRCOPENTIME="01.01.2007 10:00:00" PRCNAME="Выполнение работ
по ремонту кровли " PRCSTATUS="1" PRCPRICE="3734038.95" DOCNUM="133"
SECONDPARTEVALDATE="17.04.2017" ACTNREPORTDATE="18.04.2017 14:15:00"
PRCYEAR_FACT="2017">
        <ROWDATA TABLE_GUID="6737824ABC97887842D25A5B857D7B6C" COUNT="9">
                <ROW IDPRCREQUIREMENTS="84616" IDREQUIREMENTS="20" IDPRC="37319"
PART="2" />
                <ROW IDPRCREQUIREMENTS="84617" IDREQUIREMENTS="21" IDPRC="37319"
PART="2" />
                <ROW IDPRCREQUIREMENTS="84619" IDREQUIREMENTS="23" IDPRC="37319"
PART="2" />
        </ROWDATA>
        <ROWDATA TABLE_GUID="FF4B30169F96A4CB419A636A868BD28E">
                <ROW IDPRCPRIVILEGESPR="11716" IDPRC="37319" IDPRIVILEGESPR="1"
GUID="6EA4BA64DD61BB4C4DBD86C167BD4255"/>
        </ROWDATA>
        <ROWDATA TABLE_GUID="708A422B45FB8D5F4C4A9A0EF15BA764">
                <ROW IDRQRTPRC="28923" IDPRC="37319" IDRQRT="23806183"
GUID="1F18D27FA517BDAC4A681448FB7CB20E"
IDRQRT_GUID="1356CCDD7FB380A144C04B96124C82E5" LOTNUM="1" />
        </ROWDATA>
</ROW>
```

FIGURE II. THE FRAGMENT OF XML-FILE: EXAMPLE 1

Figure 3 shows the fragment of XML-file where the record attributes are specified by the set of tables (example 2). First, it specifies the parent table which lists all records and then it specifies one common property table which lists all property records with references to parent records. Starting with the code <*ROWDATA TABLE_GUID = "87362F010B3DBE1747E06A8FEC967831" COUNT = "5853"*> the parent records are listed; starting from the code <*ROWDATA TABLE_GUID = "FF4B30169F96A4CB419A636A868BD28E" COUNT = "5940"*> the property records are listed. Each of the property records refers to the parent record.



```
<ROWDATA TABLE_GUID="87362F010B3DBE1747E06A8FEC967831" COUNT="5853">
        <ROW IDRQRT="4681" RQRT="Проживание в летних (сезонных) детских лагерях."
NOTICEPUBLDATE="01.01.2013" IDPRCACTIVITY="10" IDMO="951"
GUID="1FB534C639A8A69B4182492C8041CB62" CODEBUDGET="262"/>
        <ROW IDRQRT="4682" RQRT="Подарочные наборы и сувенирные изделия."
NOTICEPUBLDATE="01.01.2013" IDPRCACTIVITY="10" IDMO="951"
GUID="766BB2C5436C9E9340EC609FF243D351" CODEBUDGET="290"/>
        <ROW IDRQRT="4683" RQRT="Подарочные наборы и сувенирные изделия."
NOTICEPUBLDATE="01.01.2013" IDPRCACTIVITY="10" IDMO="951"
GUID="77A4C783BE4BA5144359479D3DD5F431" CODEBUDGET="290"/>
        ...
</ROWDATA>
<ROWDATA TABLE_GUID="FF4B30169F96A4CB419A636A868BD28E" COUNT="5940">
        <ROW IDRQRT_BDGT="257404" IDRQRT="19459" IDINCOME="2" IDEXPENSE="1" IDGCP="1"
KBK="91710060410075130244" Y2016="93" GUID="B541EDD93E768C874B7FF771D6E6E91D"/>
        <ROW IDRQRT_BDGT="253089" IDRQRT="16475" IDINCOME="3" IDEXPENSE="1" IDGCP="1"
KBK="91710020410151611226" Y2015="4,9" GUID="8EA89BED35D2BC6B4083293133F96CC7"/>
        <ROW IDRQRT_BDGT="256628" IDRQRT="19100" IDINCOME="1" IDEXPENSE="1" IDGCP="1"
KBK="91710060467513244340" Y2015="60" GUID="8F3732437160B4EA4FAEFCFA37AC207C"/>
        ...
</ROWDATA>
```

FIGURE III. THE FRAGMENT OF XML-FILE: EXAMPLE 2

Figure 4 shows the fragment of a XML-file where the record attributes are specified as the nested tags (example 3). Moreover, the property table is located on the tag of this record. The parent record is limited by the tag <position> and the properties table <products> with several records <product> is located inside the parent record.

```
<position>
        <commonInfo>
                <positionNumber>П44201603192000254002000077</positionNumber>
                <orderNumber>77</orderNumber>
                <contractSubjectName>На право поставки автомобильных шин для нужд КГБУЗ ККПТД №
1 в 2016 году для субъектов малого предпринимательства, социально ориентированных не коммерческих
организаций. </contractSubjectName>
                <contractMaxPrice>467825.67</contractMaxPrice>
                <payments>467825.67</payments>
                <contractCurrency>
                        <code>RUB</code>
                        <name>Российский рубль</name>
                </contractCurrency>
                <placingWay>
                        <code>EA44</code>
                        <name>Электронный аукцион</name>
                </placingWay>
                <positionPublishDate>2016-02-26T12:04:15.624+07:00</positionPublishDate>
        </commonInfo>
        <products>
                <product>
                        <OKPD2>
                                <code>22.11.11.000</code>
                                <name>Шины и покрышки пневматические для легковых автомобилей
новые</name>
                        </OKPD2>
                        <name>На право поставки автомобильных шин для нужд КГБУЗ ККПТД № 1 в
2016 году для субъектов малого предпринимательства, социально ориентированных не коммерческих
организаций. </name>
                        <minRequirement>Согласно ТЗ</minRequirement>
                        <OKEI>
                                <code>642</code>
                                <name>Единица</name>
                        </OKEI>
                        <sumMax>467825.67</sumMax>
                        <price>467825.67</price>
                        <quantity>1.00</quantity>
                        <quantityCurrentYear>1.00</quantityCurrentYear>
                </product>
                <product>
                        …
                </product>
        </products>
</position>
```

FIGURE IV. THE FRAGMENT OF XML-FILE: EXAMPLE 3

The investigation of various forms of data representation in XML format has shown that the proposed unified model structure takes into account the features of data representation and the structure of the XML-file can be standardized as a hierarchy of its elements.

III. Transformation of Information Object into Unified Hierarchical Model

The procedure of transformation of information object into unified model contains two basic stages: checking the compatibility of the information object and the unified model and filling the elements of unified model.

The algorithm of checking the compatibility of the information object and the unified model performs the preliminary control of the format *‘rootElement.attribute ("Format")’, the direction ‘rootElement.attribute ("Direction")’* and the correspondence of data to the declared schema *‘(packageModel.validator.isCorrect (rootElement.item [1])’*. If the conditions are not met the processing is interrupted and an error message is generated. The algorithm of checking the compatibility of the information object and the unified model is shown in Figure 5.

```
Function GetPackageModel(rootElement,)
    input: rootElement: xmlRootNode;
    output: packageModel;
  begin
    if rootElement.name = "Package" then
      if rootElement.attribute("Format") <> xmlFormat then
      begin
        toReport("Cannot apply this method to current package");
        return null;
      end;
    if rootElement.attribute("Direction") not in packageInputDirections then
      begin
        toReport("Cannot apply this method to output package");
        return null;
      end;
    if packageModel.scheme is not null then
      begin
        if not packageModel.validator.isCorrect(rootElement.item[1]) then
          begin
            toReport("XML data is not valid for current package");
            return null;
          end;
      end;
    …
    packageModel.fillHeader(rootElement);
    packageModel.header.FillChilds(rootElement);
    packageModel.GUID = GenegateGUID;
    return packageModel;
  end;
```

FIGURE V. THE ALGORITHM OF CHECKING THE COMPATIBILITY OF THE INFORMATION OBJECT AND THE UNIFIED MODEL

In case of successful primary control, the processes of downloading the header of the information object ‘*packageModel.fillHeader (rootElement)*’ and recursive filling of the hierarchical model elements ‘*packageModel.header.FillChilds (rootElement)*’ are started. After loading the data the GUID is generated ‘*packageModel.GUID = GenerateGUID*’ which uniquely identifies the model for further processing. At the second stage the algorithm recursively fills the hierarchical model elements: firstly, the top-level elements ‘*fillChild (XMLElement, keys)*’, then their children ‘*fillChild (XMLElement)*’. The algorithm of filling the unified model elements is shown in Figure 6.

During the filling process, each element of the hierarchical model is assigned a GUID ‘*table.GUID = GenegateGUID*’ which allows to identify the data from the source and from the receiver. In spite of the fact that during the filling of the child elements the groups type are specified (e.g. ‘*typesValue*’, ‘*typeRecord*’, ‘*typeTable*’) the algorithm takes into account different subgroups (e.g. ‘*typeTableProperty*’, ‘*typeTableLink*’) and applies the different ways for key fields processing (e.g. *key*, *keyFields*).

The representation of information exchange object in form of a unified hierarchical model makes it possible to matching the data of different systems by operating with unified structural elements, to set the conditions for data exchange between interacting systems and to automate the transformation of one format to another without reprogramming.

```
Procedure modelElement.fillChilds(XMLElement)
        input: XMLElement: xmlNode;
    begin
      for i = 1 to self.childCount do
        begin
          currentElement = self.child[i];
          currentXMLElement = XMLElement.child[i];
          keys = currentElement.getKeys;
          currentElement.fillChild(currentXMLElement, keys);
        end;
    end;


Procedure modelElement.fillChild(XMLElement, keys)
        input: XMLElement: xmlNode;
                [keys = null];
    begin
      switch self.typeGroup;
        case typeValue:
        if not CheckschemeRestrictions(XMLElement) then
            begin
              toReport("Incorrect value" + XMLElement.name);
              break;
            end;
            value = self.addValue(XMLElement);
            value.GUID = GenegaterGUID;
            break;
        case typeRecord:
        if not CheckschemeRestrictions(XMLElement) then
            begin
              toReport("Incorrect value" + XMLElement.name);
              break;
            end;
            record = self.addRecord(XMLElement);
            record.GUID = GenegaterGUID;
            record.keyfield = GetRecordKeyField(XMLElement, keys);
            record.fillChilds = (XMLElement, record, record.keyField);
            break;
        case typeTable:
        if not CheckschemeRestrictions(XMLElement) then
            begin
              toReport("Incorrect value" + XMLElement.name);
              break;
            end;
            keyFields = GetTableKeyFields(XMLElement);
            table = self.addTable(XMLElement);
            table.GUID = GenegateGUID;
            table.keyField = keyFields[1];
            table.fillChilds(XMLElement, table, keyFields);
            break;
        end;
        end;
        end;
    end;
```

FIGURE VI. THE ALGORITHM OF FILLING THE UNIFIED MODEL ELEMENTS

IV. CONCLUSION

The paper presents an unconventional approach to standardised data exchange between heterogeneous resources based on a unified hierarchical model of the data representation. The paper gives a metamodel and description of the structural elements of the unified model. Some conversion algorithms were developed to turn the information storage schemes into the unified hierarchical model in the case of XML-format. Unified representation of information exchange structures of various formats makes it possible to automatise the matching of information storage schemes of items in different subsystems.

ACKNOWLEDGMENT

The reported study was funded by Russian Foundation for Basic Research, Government of Krasnoyarsk Territory, Krasnoyarsk Region Science and Technology Support Fund to the research project No 16-47-243020.

REFERENCES

[1] G. Hohpe, B. Wolf, “Enterprise integration patterns: designing, building, and deploying messaging solutions”, Addison-Wesley Professional: Computers, 2004, 683 p.

[2] D. Braue, “The Integration Imperative”, 2002, // http://www.cio.com.

[3] B. Hochgurtle, “C# and Java: cross-platform Web-services”, Svjas, 2004, 213 p. (in Russian)

[4] J. Dědič, “Advanced Topics on System Integration”, Jiří Dědič, Masaryk University, Brno, 2005, 108 p.

[5] T. Penkova, A. Korobko, A.Belorusov,“Development of a unified model of data representation for cross-system interaction”, Advances in Intelligent Systems Research, 2016, vol.133, pp. 48-51.

[6] M. F. Fernandez, A. Morishima, D. Suciu, W.-Ch. Tan, “Method for converting relational data into XML”, patent US 6785673 B1, 2004.

[7] B. Vrdoljak, M. Banek S. Rizzi, “Designing Web Warehouses from XML Schemas”, LNCS, vol. 2737, 2003, pp. 89-98.

[8] E. Rahn and P. A. Bernstein, “A survey of approaches to automatic schema matching”, Very Large Database J., 10(4), 2001, pp. 334-350.

[9] "Contract system in the procurement of goods, works and services for state and municipal needs": The Federal Law N 44-FZ, 05.04.2013 (ed. from 03.07.2016).

[10] R. V. Morozov, “Consolidation of procurement planning functions in a single system “Municipal customer”, Proceedings of XIV Russian conference “Problems of Region informatization”, Krasnoyarsk, 2015, p.161-165 (in Russian).