DOI: 10.14529/ctcr170118

IDEF0-DIAGRAM INTO DATABASE CONVERSION APPROACH DEVELOPMENT

A.A. Vakalyuk, *avakalyuk@yandex.ru*,
S.N. Basmanov, *seregabasmanov@rambler.ru*
Ural State University of Railway Transport, Ekaterinburg, Russian Federation

An approach was developed to the conversion IDEF0-diagram, created in the business processes design tool CA ERwin Process Modeler, to database by analysis of *.XML file for task solution of organization business processes modeling within the scope of information system. Due to recognizing names and links between functional blocks, their decompositions and interface connections the analysis of the basic tags of *.XML file was done. Based on the analysis *.XML file database structure is developed, in which all the tables are related by "one-to-many" with a foreign key. Research results display actual tasks for companies. Solving tasks companies can move to a new technological and organizational level and compete more effective in modern economic conditions. Job was done by 05.13.01 specialty – systems analysis, control and information processing (branches).

*Keywords: IDEF0-diagram, information system, *.XML file, functional blocks, interface arrows.*

Introduction

One of the basic stages of information systems (IS) development is a modeling of business processes of the organization. CA Erwin Process Modeler is one of the popular design tools of business processes. This tool uses methodology of the IDEF0. IDEF0 is a system's functional model, which describes a complex of system functions and defines system morphology (its structure) – composition of the subsystems, their relationship. This model regards the system as a set of operations; each of them converts some object or collection of objects [1].

In this article a new approach to the conversion IDEF0-diagram into MySQL database (DB) by analysis of *.XML file is proposed.

This approach is used to include business processes diagram data to IS as a structural part with subsequent distribution between departments of company.

Purpose of the research is approach development to the conversion IDEF0-diagram to DB by analysis of *.XML file.

1. *.XML file analysis

The first stage of conversion IDEF0-diagram to DB is diagram creation in the design tool of business processes – CA Erwin Process Modeler. In this case, the diagram stored in an *.XML file.

Example A-0 diagram shown in Fig. 1.

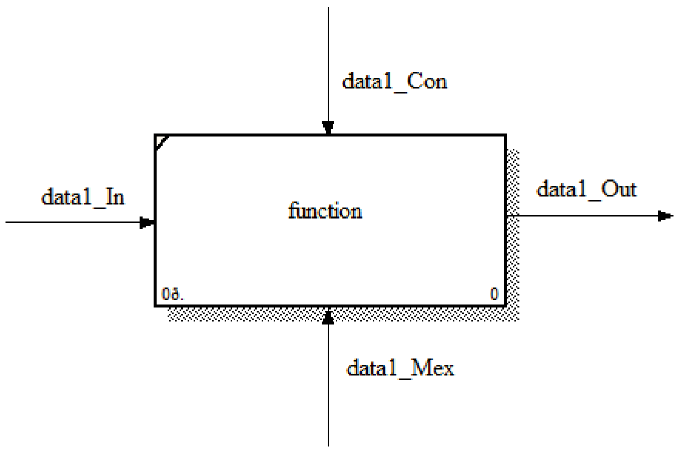

Fig. 1. IDEF A-0 diagram

The next stage is recognition of name and links between functional blocks and interface arrows in *.XML file.

Example *.XML file cutting with tags is shown below.

```
<ProcessModeler>
  <PMModel id="…" Name="IDEF-diagram">
    .
    .
    .
    <PMArrow_Groups>
      <PMArrow id="…" Name="Data1_In">...</PMArrow>
      <PMArrow id="…" Name="Data1_Out">...</PMArrow>
      <PMArrow id="…" Name="Data1_Con">...</PMArrow>
      <PMArrow id="…" Name="Data1_Mex">...</PMArrow>
    </PMArrow_Groups>
    <PMActivity_Groups>
      <PMActivity id="…" Name="function">...</PMActivity>
    </PMActivity_Groups>
    <PMDiagram_Groups>
      <PMDiagram id="…" Name="Function">
        <PMDiagramProps>
          .
          .
          .
          <PMParentDiagramRef>…</PMParentDiagramRef>
          .
          .
          .
          <PMTopRef>…</PMTopRef>
          <PMBottomRef>…</PMBottomRef>
          <PMLeftRef>…</PMLeftRef>
          <PMRightRef>…</PMRightRef>
        </PMDiagramProps>
          .
          .
          .
        <PMBox_Groups>
          <PMBox id="…" Name="">
            <PMBoxProps>
              <PMBoxCoordinates member_name="left">… </PMBoxCoordinates>
              <PMBoxCoordinates member_name="top">… </PMBoxCoordinates>
              <PMBoxCoordinates member_name="right">… </PMBoxCoordinates>
              <PMBoxCoordinates member_name="bottom">… </PMBoxCoordinates>
              <PMActivityRef>…</PMActivityRef>
              .
              .
              .
            </PMBoxProps>
          </PMBox>
        </PMBox_Groups>
        <PMArrowLabel_Groups>
          <PMArrowLabel id="…" Name="">
            <PMArrowLabelProps>
              .
              .
              .
              <PMArrowRef>…</PMArrowRef>
              .
              .
              .
            </PMArrowLabelProps>
            <PMArrowSegment_Groups>
              <PMArrowSegment id="…" Name="">
                <PMArrowSegmentProps>
                  <PMVertices index="0">(…,…)</PMVertices>
                  <PMVertices index="1">(…,…)</PMVertices>
                  <PMVertices index="2">(…,…)</PMVertices>
                  .
                  .
                  .
                  <PMSourceConnectionRef>…</PMSourceConnectionRef>
                  <PMSinkConnectionRef>…</PMSinkConnectionRef>
                  .
                  .
                  .
                </PMArrowSegmentProps>
              </PMArrowSegment>
            </PMArrowSegment_Groups>
          </PMArrowLabel>
          <PMArrowLabel id="" Name="">...</PMArrowLabel>
          <PMArrowLabel id="" Name="">...</PMArrowLabel>
          <PMArrowLabel id="" Name="">...</PMArrowLabel>
        </PMArrowLabel_Groups>
      </PMDiagram>
    </PMDiagram_Groups>
  </PMModel>
</ProcessModeler>
```

IS, from organizational point of view, is a difficult process, which can be represented by several models. In this case, it is necessary to take into account the possibility of the new models addition and differentiate between the existent models in the system. For this, analysis of a new *.XML file is necessary to begin with determination of model's id and Name, which are situation within the tag <PMModel id="…" Name="…">, and comparison them with existing. If such model is already exists in the system, then old model in the system is deleted and is replaced by new model.

Information about interface arrows groups, which are used in this model, is presented within the tag <PMArrow_Groups>… </PMArrow_Groups>. Description of arrows' properties and identification of each arrow are presented within tag <PMArrow id="…" Name="…">...</PMArrow>. In this tag it is assigned unique id and Name for every interface arrow. "Name" conforms the "Arrow Name" of interface arrow, which user sets in process diagram creation by program CA Erwin Process Modeler.

Information about functional blocks groups, which are used in this model, is presented within the tag <PMActivity_Groups>… </PMActivity_Groups >. The description of functions and identification of each blocks are presented within tag <PMActivity id="…" Name="…">...</PMActivity>. This functional block is assigned unique id and Name. "Name" is the name of the user-defined in "Activity Properties" for functional block at the stage of diagram creating in the program CA Erwin Process Modeler.

One of the main features of IDEF0-diagram is a principle of compound functional blocks decomposition on simpler blocks on the new diagram. The new diagram is a child diagram, and its functional blocks are child blocks, which present subfunctions of the parent-block of parent-diagram. Each of functions of the child diagram can be detailed by decomposition of the corresponding functional block [2].

All parent and child diagrams are presented within tag <PMDiagram_Groups>… </PMDiagram_Groups>. Each of them has own unique id and Name, which are situated within tag <PMDiagram id="…" Name="…">…</PMDiagram>.

Also, the tag <PMDiagram>…</PMDiagram> contains such tags as <PMDiagramProps>... </PMDiagramProps>, <PMBox_Groups>… </PMBox_Groups> and <PMArrowLabel_Groups>... </PMArrowLabel_Groups>.

The structural feature of child diagram is the presence of the tag <PMParentDiagramRef>…

</PMParentDiagramRef>, in which is shown parent diagram id. This tag is situated within tag <PMDiagramProps>...</PMDiagramProps>.

Input and output from process objectives are presented on IDEF0-diagram by interface arrows, can be directed from one functional block to another block, or only to functional block. In this case, when interface arrows are directed only to block, they can come from upper, bottom, right and left boundaries of diagram. These boundaries are presented in the tags <PMTopRef>…</PMTopRef>, <PMBottomRef>…</PMBottomRef>, <PMLeftRef>…</PMLeftRef>, <PMRightRef>… </PMRightRef> within <PMDiagramProps>... </PMDiagramProps>.

The functional blocks within the diagram are described within the tag <PMBox_Groups>...</PMBox_Groups>. Each of the functional blocks is assigned id in the tag <PMBox id="…" Name="">…</PMBox>.

Each of the four sides of the functional block has a certain value (the role) and defines the type of interface, i.e. a way of interaction of arrow with block: the upper side means “Control”, the left side means “Input”, the right side means “Output”, the bottom side means “Mechanism” [3]. In the *.XML file each of the functional block sides is assigned the numerical value received relative to the origin of coordinates, located in the upper left corner. This numerical values are assigned for left, upper, right and bottom sides within the following tags: <PMBoxCoordinates member_name="left"> … </PMBoxCoordinates>, <PMBoxCoordinates member_name="top"> … </PMBoxCoordinates>, <PMBoxCoordinates member_name="right"> … </PMBoxCoordinates>, <PMBoxCoordinates member_name="bottom"> … </PMBoxCoordinates> accordingly. These tags are situated within the tag <PMBoxProps>...</PMBoxProps>.

For the functional block identification, within tag <PMBoxProps>...</PMBoxProps> also is situated the tag <PMActivityRef>…</PMActivityRef>, in which text coincides with id, represented in the tag <PMActivity>...</PMActivity>.

Thereby, analyzing the content of the tag <PMBox_Groups> ... </PMBox_Groups>, the name and number of the functional blocks and their location on the diagram can be identified.

The interface arrows being represented within the scope of the diagram are described within the tag <PMArrowLabel_Groups>...</PMArrowLabel_Groups>. The unique id, which is assigned within the tag <PMArrowLabel id="…" Name="">… </PMArrowLabel>, is used for identification of every arrow. This tag contains a number of features pertaining to the arrow and being situated within the tags: <PMArrowLabelProps>...</PMArrowLabelProps> and <PMArrowSegment_Groups>... </PMArrowSegment_Groups>.

The first tag contains the dependent tag <PMArrowRef>…</PMArrowRef>, which contains the id, which is identical to text within the tag <PMArrow>...</PMArrow>.

Thereby, one can identify the interface arrow’s name, which is used within the scope of the diagram.

The tags <PMSourceConnectionRef>…</PMSourceConnectionRef>, <PMSinkConnectionRef>… </PMSinkConnectionRef> are used for direction identification of interface arrows. These tags indicate to the start and the end of interface arrows accordingly and are situated within the tag <PMArrowSegment_Groups>...</PMArrowSegment_Groups>. The text within <PMSourceConnectionRef>… </PMSourceConnectionRef> can coincide with the text of document border or with id of functional block <PMBox>…</PMBox>, from which the arrow issues.

Also, the text within <PMSinkConnectionRef>…</PMSinkConnectionRef> can coincide with text of document border or with id of functional block <PMBox>…</PMBox>, from which the arrow enters.

There are five types of interface arrows in IDEF0: input, output, control, mechanism and call. For the arrows type identification the tags <PMVertices index="0">(…,…)</PMVertices>, <PMVertices index="1"> (…,…)</PMVertices>…<PMVertices index="n">(…,…)</PMVertices> are used in *.XML file and they are situated within the tag <PMArrowSegment_Groups>... </PMArrowSegment_Groups>. The number “n” indicates a quantity of interface arrow coordinates, and can vary depending on bends. The interface arrow coordinates description is (x_0, y_0), (x_1, y_1), (0, 1 or 2), (x_2, y_2), (0, 1 or 2) etc., where x_n – indent from origin axle x, y_n – indent from origin axle y, (0, 1) – horizontal direction of arrow, (0, 2) – vertical direction of arrow. If the end tag has coordinates (0, 1), that the arrow is an input or output. If the end tag has coordinates (0, 2), that the arrow is one of three types: mechanism, control and call. The interface arrow’s starting point has a coordinate (x_0, y_0), and end point – (x_{n-1}, y_{n-1}).

For more detailed identification of arrow type, it is checked the location of the starting point coordinate and end point coordinate on the functional block sides, which are shown the numeric coordinates within tags: <PMBoxCoordinates member_name="left"> … </PMBoxCoordinates>, <PMBoxCoordinates member_name="top"> … </PMBoxCoordinates>, <PMBoxCoordinates member_name="right"> … </PMBoxCoordinates>, <PMBoxCoordinates member_name="bottom">…</PMBoxCoordinates>. The type of arrow is identified depending on the value of the functional block side.

Thereby, analyzing contents of the tag <PMArrowLabel_Groups>...</PMArrowLabel_Groups>, one can identify names and a number of interface arrows of diagram and cooperation of arrows with functional blocks.

Thereby, after analysis the *.XML file, relationship between child and parent diagrams, functional blocks and their decompositions, a number of functional blocks and a number of interface arrows in the scope of the diagram can be detected.

2. DB structure development

On the next stage, received from *.XML file data are recorded in MySQL DB. The structure of DB is shown on Fig. 2.

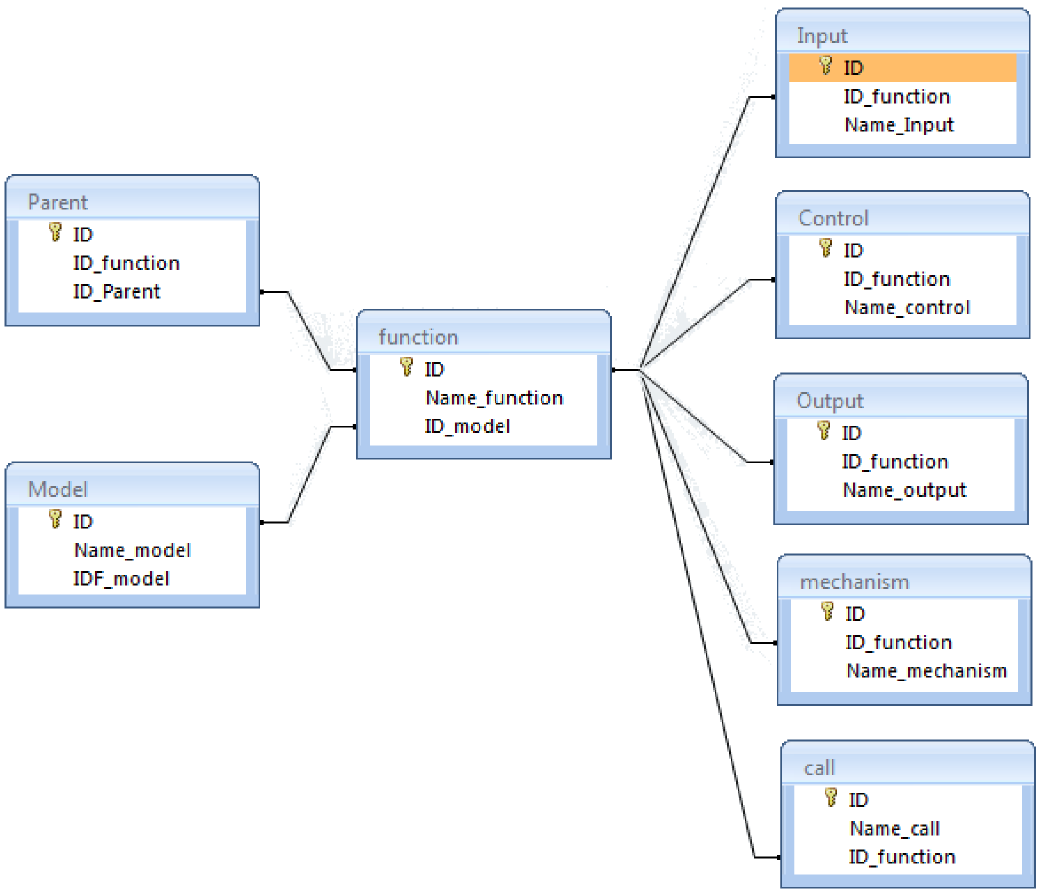

Fig. 2. The IDEF0-diagram DB structure

The main feature of this stage is a realization of relation “one-to-many” between tables by foreign keys. Foreign key is a data column of one table and coincide with primary key (or its part) of another table [4]. The realization of this relation makes it possible to minimize data redundancy and to ensure data integrity.

The DB consists of eight related tables. Addition new or edition existent IDEF0-diagrams is carried out in a table “Model”. The model’s name (Name_model) and identification number (IFD_model) assigned by CA Erwin Process Modeler are recorded in this table. Attribute AUTO_INCREMENT of column “ID” is used for new rows generation identifier.

The basic table of DB is a table “function”. It consists of the functional blocks’ names (Name_function) of IDEF0-diagram. The link between tables “function” and “model” is carried out by foreign key (ID_model) and primary key (ID) accordingly.

The identifier (ID_Parent) of functional block and its decompositions is located in a table “Parent”.

The interface arrows of IDEF0-diagram are represented in tables “input”, “output”, “mechanism” and “call”. Each of these tables consists of three columns: identification number of interface arrow, its name and foreign key, which conforms to primary key of table “function”.

The main idea of use of foreign key is use of expressions ON UPDATE CASCADE and ON DELETE CASCADE. Thanks them one can automatically update and delete connection rows of all tables of DB.

Example table “function” creation, using foreign key:

```
CREATE TABLE function(
ID INT(11) NOT NULL AUTO_INCREMENT,
Name_function varchar(60),
ID_model INT(11) NOT NULL,
PRIMARY KEY (ID),
FOREIGN KEY (ID_model) REFERENCES model(ID)
ON UPDATE CASCADE
ON DELETE CASCADE) ENGINE = InnoDB;
```

Thereby, DB structure development makes it possible to minimize data redundancy and to ensure data integrity.

Conclusions

1. An approach was purposed to the conversion IDEF0-diagram, created in the business processes design tool CA ERwin Process Modeler, to database by analysis of *.XML file. This enable to include of business processes diagram data in IS as a structural part with subsequent distribution between departments of company.

2. The developed database structure makes it possible to minimize data redundancy and to ensure data integrity.

3. The realization of this approach will enable to transfer organization on new technology and organizational level and make it more competitive in modern economic conditions.

References

1. Samujlov K.E., Serebrennikova N.V., Chukarin A.V., Jarkina N.V. *Osnovy formal'nyh metodov opisanija biznes-processov: ucheb. posobie* [Basic Foundation of Formal Methods of Business Processes Description: Tutorial]. Moscow, RUDN, 2008. 130 p.

2. Kuljabov D.S., Korol'kova A.V. *Vvedenie v formal'nye metody opisanija biznes-processov*: *ucheb. posobie* [An Introduction to Formal Methods of Business Processes Description: Tutorial]. Moscow, RUDN, 2008. 173 p.

3. Gorbachenko V.I., Ubiennyh G.F., Bobrysheva G.V. *Sozdanie funkcional'noj modeli informacionnoj sistemy s pomoshh'ju CASE-sredstva CA ERwin Process Modeler 7.3*: *ucheb. posobie* [Functional Model Creation of Information System by CASE-Tool CA ERwin Process Modeler 7.3]. Penza, PGU, 2010. 66 p.

4. Orlov S.A. *Programmnaja inzhenerija: uchebn. dlja vuzov. 5-e izdanie obnovlennoe i dopolnennoe. Standart tret'ego pokolenija* [Software Engineering: Tutorial for Institutes of Higher Education. The 5-th Updated and Supplemented Edition. Standard of the Third Generation]. St. Petersburg, Piter, 2016. 640 p.

Received 1 November 2016

УДК 004.9 **DOI: 10.14529/ctcr170118**

РАЗРАБОТКА ПОДХОДА К КОНВЕРТИРОВАНИЮ IDEF0-ДИАГРАММЫ В БАЗУ ДАННЫХ

А.А. Вакалюк, С.Н. Басманов
Уральский государственный университет путей сообщения, г. Екатеринбург

Разработан подход к конвертированию IDEF0-диаграммы, созданной в средстве проектирования бизнес-процессов CA ERwin Process Modeler, в базу данных по средствам анализа *.XML файла для решения задачи моделирования бизнес-процессов организации в рамках информационной системы. Произведен анализ основных тегов *.XML файла с целью распознания имен и связей между функциональными блоками, их декомпозициями и интерфейсными связями. На основе анализа *.XML файла разработана структура БД, в рамках которой все таблицы связаны отношением «один-ко-многим» при помощи внешнего ключа. Полученные в ходе исследования результаты отражают актуальные задачи, стоящие перед организациями, реализация которых позволит перевести предприятие на новый технологический и организационный уровни, и сделать его более конкурентоспособным в современных экономических условиях. Работа выполнена по специальности 05.13.01 – Системный анализ, управление и обработка информации (по отраслям).

*Ключевые слова: IDEF0-диаграмма, информационная система, *.XML файл, функциональные блоки, интерфейсные стрелки.*

Литература

1. *Основы формальных методов описания бизнес-процессов: учеб. пособие / К.Е. Самуйлов, Н.В. Серебренникова, А.В. Чукарин, Н.В. Яркина. – М.: РУДН, 2008. – 130 с.*
2. *Кулябов, Д.С. Введение в формальные методы описания бизнес-процессов: учеб. пособие / Д.С. Кулябов, А.В. Королькова. – М.: РУДН, 2008. – 173 с.*
3. *Горбаченко, В.И. Создание функциональной модели информационной системы с помощью CASE-средства CA ERwin Process Modeler 7.3: учеб. пособие / В.И. Горбаченко, Г.Ф. Убиенных, Г.В. Бобрышева. – Пенза: ПГУ, 2010. – 66 с.*
4. *Орлов, С.А. Программная инженерия: учеб. для вузов / С.А. Орлов. – 5-е изд. обновл. и доп. – СПб.: Питер, 2016. – 640 с. – (Стандарт третьего поколения).*

Вакалюк Андрей Александрович, канд. техн. наук, доцент кафедры мехатроники, Уральский государственный университет путей сообщения, г. Екатеринбург; avakalyuk@yandex.ru.

Басманов Сергей Николаевич, аспирант кафедры мехатроники, Уральский государственный университет путей сообщения, г. Екатеринбург; seregabasmanov@rambler.ru.

Поступила в редакцию 1 ноября 2016 г.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Vakalyuk, A.A. IDEF0-Diagram into Database Conversion Approach Development / A.A. Vakalyuk, S.N. Basmanov // Вестник ЮУрГУ. Серия «Компьютерные технологии, управление, радиоэлектроника». – 2017. – Т. 17, № 1. – С. 153–159. DOI: 10.14529/ctcr170118

FOR CITATION

Vakalyuk A.A., Basmanov S.N. IDEF0-Diagram into Database Conversion Approach Development. *Bulletin of the South Ural State University. Ser. Computer Technologies, Automatic Control, Radio Electronics*, 2017, vol. 17, no. 1, pp. 153–159. DOI: 10.14529/ctcr170118