Semi-Automated Formalization and Representation of Engineering Knowledge Extracted from Spreadsheet Data

Aleksandr Yu. Yurin¹, Nikita O. Dorodnykh¹, and Alexey O. Shigarov¹

¹Matrosov Institute for System Dynamics and Control Theory, Siberian Branch of the Russian Academy of Sciences, 134 Lermontov St., Irkutsk, 664033, Russia

Corresponding author: Aleksandr Yu. Yurin (e-mail: iskander@icc.ru).

This work was supported by the Russian Science Foundation, grant number 18-71-10001.

ABSTRACT The development of new methods and tools for formalization and representation of complex knowledge in the context of the creation of intelligent systems remains in the scope of scientific research. There is a trend to automate the knowledge formalization and representation by using various sources of information, in particular, spreadsheets. This paper proposes a novel approach to the semi-automatic formalization and representation of engineering knowledge in the form of conceptual models and knowledge base codes from spreadsheet data. Our approach contains three phases: (I) rule-driven data transformation source spreadsheet tables to a specific canonical form (data preprocessing), (II) domain knowledge formalization and representation via the extraction and aggregation of conceptual model fragments from canonicalized tables, (III) model-driven synthesizing knowledge base and source codes from a domain model. The approach is implemented by our tools: TABBYXL provides the development of a software application for the spreadsheet data canonicalization; Personal Knowledge Base Designer is used to build and aggregate conceptual models fragments, as well as to construct a target knowledge base and generation source codes. Our case study on the industrial safety inspection (ISI) demonstrates the applicability of the approach for prototyping knowledge bases containing decision-making rules.

INDEX TERMS Spreadsheet, conceptual model, data transformation, domain model generation, knowledge base engineering, industrial safety inspection

I. INTRODUCTION Currently, the development of new methods and tools for formalization and representation of complex engineering knowledge in the context of the creation of domain-specific knowledge-based systems for solving various practical engineering problems remains in the scope of scientific research [1]. Such systems are actively used in design [2], diagnostics and forecasting conditions of complex technical systems [3], selection of design materials [4], risk analysis [5], etc.

The main elements of intelligent systems are knowledge bases, the research of which does not lose its relevance [6]. In recent years, there has been an increasing trend to automate knowledge formalization and representation by using a semi-automated transformation of various sources of information (texts, documentation, databases, spreadsheet tables, web resources, etc.), as well as the principles of visual, conceptual and cognitive modeling.

Spreadsheets are one of the most convenient ways to structure and represent statistical and other data. For this reason, they are widely distributed, and their number, only in the Internet, is about 150 millions [7]. Spreadsheets contain useful knowledge in many domains (e.g., engineering, business, etc.), and can be a valuable knowledge source. In the last decade, approaches to automating the development of knowledge bases have emerged based on the analysis of spreadsheets in CSV and Excel format, as well as web tables [8-9]. These approaches focus on automated extracting knowledge from tabular data and, as a rule, they focus on a specific structure (model) of the table. Moreover, raw tabular data are not always containing the experience and expert knowledge that are important when solving subject tasks. Besides, knowledge bases require a detailed description, confirmation, and verification by experts. In this regard,
ontology and conceptual modeling methods are used in addition to build complete and consistent knowledge bases, when the spreadsheets are a source of knowledge at the terminological and essential levels.

This paper presents a novel three-phase approach to a semi-automatic formalization and representation of engineering knowledge in the form of conceptual models and spreadsheet data, and covers the following tasks:

(I) extracting source arbitrary tables from a spreadsheet and transforming them to an original canonicalized form;

(II) formalization and representation domain model fragments extracted from canonicalized tables and aggregating them into a complete domain model;

(III) synthesizing source codes of a knowledge base in a target knowledge representation language from a complete domain model.

We develop and integrate our two tools to demonstrate the applicability of the proposed approach:

(I) TabbyXL [10] for canonicalization of source arbitrary tables to a canonicalized form;

(II) PKBD (Personal Knowledge Base Designer) [11] for formalization and representation of knowledge in the form of domain models and source codes of knowledge bases.

So, our contribution consists of the following results. For the first time, we proposed the approach for formalization and representation of complex engineering knowledge in the form of conceptual models and knowledge bases that utilizes the rule-driven spreadsheet data extraction. Our approach facilitates formalization, representation, and codification of knowledge and data represented in the form of spreadsheets.

The experiments were conducted to demonstrate that the approach is a feasible way for generating fragments of a conceptual model (ontology) in the Industrial Safety Inspection (ISI) area (it is our case study). As a result, conceptual models for ISI tasks were developed. It is important to highlight that the resulting conceptual models were used to synthesize source codes and high-level specifications for knowledge bases. We also showed such synthesis by an example of the generation of knowledge base represented as a source code in CLIPS (C Language Integrated Production System), DROOLS, and PHP (Hypertext Preprocessor).

The paper is organized as follows. Section 2 considers the related works. Section 3 presents the preliminaries of this work. Section 4 explains our three-phase approach including a description of the implementation. Section 5 illustrates a case study of our approach for solving ISI task including an illustrative example and the experimental evaluation, Section 6 discusses the results, while Section 7 presents concluding remarks and future work.

II. RELATED WORKS

Analyzing and transforming data from spreadsheets to formalize and represent knowledge is a popular area of research. At the same time, solutions in this area can be divided into two large groups of approaches that implement:

- end-to-end or full transformation [12-18]: provide transformation of source spreadsheets directly into some knowledge structures, in most cases it is the Resource Description Framework (RDF) or Web Ontology Language (OWL) files;

- partial or step-by-step transformation [19-23]: provide partial solutions using intermediate forms of data and knowledge representation, for example, in the form of conceptual models; they separately solve problems of transforming spreadsheets and conceptual models (mainly in the form of UML class diagrams).

Next, let’s present some examples of these studies.

A. END-TO-END AUTOMATED TRANSFORMATION OF SPREADSHEET DATA INTO KNOWLEDGE

Currently, the most popular way to represent domain knowledge is knowledge graphs (semantic networks) or ontologies. For this reason, most of the studies aimed at obtaining machine-interpreted knowledge structures based on understanding tabular data (including solving tasks of detection, analysis, and transformation of tables) are focused on the RDF and OWL formats.

From the methodological point of view, they provide the extraction of separate knowledge fragments (or mini-ontologies) with their subsequent aggregation into one expanding domain model. An example of such a study is the TANGO approach [9]. There are also examples of research and commercial software: RDF123 [13], Owlifier [24], Datalift [16], Any2OWL [17], Spread2RDF, Any23, TopBraid Composer, etc.

The main disadvantages of end-to-end spreadsheet data transformation studies are the following:

- orientation to well-structured data, which implies using either a specific table layout (usually “entity” tables containing descriptions of instances of a particular entity, with all columns being its attributes, i.e. tables of the one dimension) or specific data sets (for example, the Gold standard);

- orientation to a specific way of presenting tables, for example, HTML;

- high qualification requirements when describing transformations (in cases where it is possible to configure systems);

- lack of validation and representation of the knowledge obtained using visual domain-specific or system-wide notations;

- lack of code generation in a specific knowledge representation language that allows one to direct use and integrate the obtained codes in applications.

Despite significant progress in this area and the growing popularity of integration of spreadsheet data and semantic technologies, the studies aimed at analyzing and

1 T2dv2 Gold Standard for Matching Web Tables to DBpedia, http://webdatacommons.org/webtables/goldstandardV2.html.
transforming spreadsheet data into other knowledge representation formalisms are poorly presented. The exception is associative rules [18], which are part of data mining methods and are not considered in these studies as an element of knowledge-based or expert systems.

In general, the trend of weak support for other knowledge representation formalisms can be explained by the possibility of their equivalent transformation; however, this does not solve the issue of code generation for a particular programming language. In turn, rule-based knowledge representation languages (such as DROOLS, JESS, CLIPS) are still useful when developing industrial intelligent and knowledge-based systems. Knowledge bases with rules attract developers with their visibility, high modularity, easy of making additions and changes, and simplicity of the logical inference mechanism.

B. PARTIALLY AUTOMATED TRANSFORMATION OF SPREADSHEET DATA INTO KNOWLEDGE

A As an example of studies that can provide partial transformation of spreadsheet data into knowledge, we can highlight studies that provide solutions of two different tasks, from which it is possible to build a step-by-step transformation chain or a pipeline:

- transforming spreadsheet data to conceptual models,
- transforming conceptual models to knowledge bases and source codes.

Although conceptual models can be used to represent ontologies and some researchers identify them with knowledge graphs and semantic networks, however, in most cases they reflect only one of the specific aspects of ontologies, for example, structural or behavioral. In this regard, there are studies aimed at obtaining certain types of conceptual models from spreadsheet data.

In particular, we can distinguish the following studies, which consider the solution of the first task.

Hung et al. [8] propose TRANSHEET, an approach for transforming spreadsheet data to a structured target model in the XML format. TRANSHEET enables users to perform mappings via a familiar and expressive spreadsheet-like formula language. This language is designed for specifying mappings between spreadsheet data and the target schema.

Hermans et al. [27] present a systematic approach called GYRO for the automation of the extraction of UML class diagrams from spreadsheet data. GYRO automatically transforms spreadsheet data in Excel format by exploiting the commonality in tables, like the two-dimensional patterns. These patterns are located within a spreadsheet table using a combination of parsing and pattern matching algorithms.

Cunha et al. [21] suggest an approach called CLASSSHEETS based on searching for functional dependencies between data cells when results of the transformation are relational models. The authors of [21] also show how to systematically transform extended CLASSSHEETS models to UML class diagrams enriched with constraints expressed in OCL (Object Constraint Language). UML class diagrams are generated under the notation of the USE framework.

Amalfitano et al. [20] describe a heuristic-based reverse engineering process for inferring conceptual data models in the form of UML class diagrams from spreadsheet tables in the Excel format. This process is fully automatic and has been defined in an industrial context and validated by an experiment involving three different spreadsheet-based information systems from the considered automotive industrial domain.

Most of the studies in this group also have some limitations related to the support of certain specific predefined models of source spreadsheets with a mixed logical and physical structure, and they are almost all focused on obtaining UML class diagrams.

The second task is usually solved independently of the first one. The existing solutions focus on analyzing specific formats of conceptual models and generating knowledge bases in the form of ontologies or logical rules presented in specific knowledge representation languages.

In particular, Zedlitz and Luttenberger [22] present an approach for transforming UML class diagrams into OWL 2 ontology using the QVT-r language from OMG standard.

Albert and Franchini [19] proposed an integrated method and ORM2OWL tool for transforming conceptual domain models in the Object Role Modelling (ORM) format into OWL ontology. eXtensible Stylesheet Language Transformations (XSLT) was used at the first stage of transforming conceptual models from XML format to ORM.

Starr and Oliveira [23] proposed a method using CmapTools conceptual maps as the main means for expressing expert knowledge, as well as a set of formal transformations applied to these maps to transform them into domain ontology in the OWL format.

Other examples of solving the task of automated transformation of UML models into ontologies are presented in [24-26].

These approaches and software have several drawbacks, in particular:

- the absence or limitation of the code generation of knowledge bases in different knowledge representation languages;
- a limited set of supported formats of conceptual models, as well as the complexity of describing the models for code generation;
- complex implementation of transformations, which, in turn, causes a variety of software used by researchers within each separate transformation (sometimes a separate software is used at each stage of the transformation).

Thus, the problem of developing methods and software for extracting and transforming data from spreadsheets with an arbitrary layout and data sets into ontologies and knowledge with subsequent validation of results by subject specialists, as well as generation of source codes and the
integration of them into applications remains an urgent task. At the same time, it is prospectively to use an intermediate representation of knowledge in the form of conceptual models that reflect the generated knowledge at the terminological level (T-Box) level.

In this paper, to overcome shortcomings mentioned, we propose a new approach to the semi-automated formalization and representation of complex engineering knowledge in the context of the creation of knowledge bases, the main features of which are the following:
- the main source of knowledge is spreadsheets with the arbitrary layout (structure);
- spreadsheet data is represented with the use of a special canonical form;
- conceptual models are an intermediate means to formalize and represent knowledge extracted from canonicalized spreadsheets and the basis for synthesizing source codes of knowledge bases.

III. PRELIMINARIES

A. SPREADSHEET SOURCE AND CANONICALIZED TABLE

We formally introduce tables and their types. A table is a grid of cells arranged in rows and columns. Such tables are used as visual communication patterns, as well as data arrangement and organization tools. In this paper, we primarily focus on spreadsheet tables, i.e. tables embedded in various documents and reports. Such spreadsheet tables typically contain data from various dimensions or named entities and are presented in the Excel format (XLSX or CSV). Below, we define elements of a spreadsheet table denoted as ST.

Spreadsheet table headings, STH, is a set of labels defining what each spreadsheet table row or/and column is about. Headings are typically in the first row or/and column in a spreadsheet table.

A spreadsheet table cell, ST[i,j] is specified with the row index i and the column index j. Spreadsheet cells hold values (entries) and are considered as atomic units in a spreadsheet table. These cells can also be empty. A spreadsheet table column STC, is a set of spreadsheet table cells lying vertically in column j of a spreadsheet table. A spreadsheet table row STR, is a set of spreadsheet table cells lying horizontally in line i of a spreadsheet table.

Such spreadsheet tables are designated as arbitrary in [10], since they may have a different layout and design style due to the specifics of domain data.

Another type of tables is a relational one. Relational tables contain high-quality relational data [30]. Wherein, relation spreadsheets can be converted into a relational model. Relational tables contain a set of entities, which could exist in rows (horizontal) or columns (vertical), the remainder of cells contain their descriptive attributes.

To represent extracted tabular data, we propose a canonical form that corresponds to relational tables by a layout. The special canonical form we use is formally defined as follows:

\[ CF = (D, R^H, C^H) \tag{1} \]

where \( D \) is a data column that contains only entries (i.e. values of a source table), \( R^H \) is a column of paths of row labels (i.e. headings addressing the values by rows), \( C^H \) is a column of paths of column labels (i.e. headings addressing the values by columns). A path of labels can express either a separation of a category into subcategories or reading order. In the examples, we use the vertical bar to denote separated labels in a path. Our approach uses the canonical form as an intermediate data representation between source spreadsheet tables and target conceptual models.

B. CONCEPTUAL MODEL AND MODEL TRANSFORMATIONS

A model is an abstraction of a system under study that makes it possible to have a better understanding of and to reason about it [31]. Models can be divided into different categories, in particular, mathematical models, graph models, etc. According to [32], models are divided into three levels, namely: conceptual models, specification models, and implementation models. A conceptual model represents concepts (entities) and relationships between them, and it is mainly built to formalize and represent the static characteristics of some system.

Recently, numerous conceptual modeling techniques have been created that can be applied across multiple disciplines to increase the user’s understanding of the system to be modeled [33]. Various visual and text notations, universal modeling languages, and standards, in particular, UML class diagrams, IDEF1x and others, are widely used in designing conceptual models.

Typical usage of conceptual models is to build various information systems, for example, knowledge-based systems. Conceptual models can be used as the basis for generating domain ontologies and knowledge bases. The model transformation aims at converting source models to target ones.

IV. APPROACH

Our approach consists of three phases: (I) extracting data from source tables and transforming them to the canonical form, (II) formalization and representation of knowledge extracted from canonicalized tables, (III) synthesizing source codes of a target knowledge base from the conceptual model. The workflow diagram of this process is shown in Figure 1.

A. OBTAINING SPREADSHEETS

The source data we are interested in can be found in document tables. First of all, such tables should be extracted from digital media like web-pages and PDF documents. Note that we do not determine this process here. However, it can
Phase 2: Formalization and representation of knowledge

Phase 3: Synthesizing rule-based knowledge bases

FIGURE 1. Diagram of the workflow for formalization and representation of engineering knowledge extracted from spreadsheet data.

be realized by using data scraping tools (e.g. Tabula or TabbyPDF) with additional manual verification and correction.

Phase 1 starts with extracting spreadsheet tables as shown in Figure 1. Tables that we are interested in typically have an arbitrary layout and not a relational one. Phase 1 aims at transforming them to the canonical form (e.g. Figure 2). We consider this process in terms of the table understanding [9] with the following steps: (I) role analysis, i.e. recovering data items (entries and labels) from cells, (II) structure analysis, i.e. recovering relationships between data items, (III) interpretation, i.e. separating labels into named or anonymous categories.

In our case, extracted tables are represented by two source forms resulted from our case study. The first of them (Form 1) have merged cells expressing either data repeating or hierarchical relationships between headings. An example of a source table in Form 1 is illustrated in Figure 2 (left). The target table in the canonical form corresponding to the source table is shown in Figure 2 (right).

To develop rules for analysis and interpretation of source tables in Form 1, we use the following assumptions. A corner cell $s$ started from 1-row and 1-column contains a stub head. The corner covers all head rows and all stub columns. A head cell contains one column label (e.g. $h_1$, $h_2$, ..., $h_n$). They can compose hierarchical (parent-child) relationships expressed by spanning and nested cells. A child label should be placed in a nested cell while its parent label is in the corresponding spanning (e.g. $h_1$ and $h_2$ are children of $h_0$ parent in Figure 2). Each path of column labels belongs to the category $C_m$ (e.g. “Column Heading” column of the canonicalized table in Figure 2). A cell of a stub part contains one row label (e.g. $s_1$, $s_2$, ..., $s_m$). A multicolumn stub expresses hierarchical paths of parent-child relationships. Each path is read in a row from a left column (parent) to the right one (child). Any path of row labels belongs to the category $R^c$ (e.g. “Row Heading” column of the canonicalized table in Figure 2).

A body part is a data block placed below the head and on the right the stub. A body cell contains one entry (e.g. $d_1$, $d_2$, ..., $d_k$). A merged cell should be considered as a set of repeated entries with the same value (e.g. $d_1$, $d_2$, and $d_k$ in Figure 2). Each entry is addressed by one path of column labels and one path of row labels. For example, $d_6$ is addressed by $(h_3|h_4)$ path of column labels, and ($s_3|s_1$) path of row labels.

These assumptions allowed us to develop a ruleset for canonicalization source tables. It was expressed in CRL, our domain-specific language for table analysis and interpretation rules [34].

It should be noted that the transformation rules depend only on the structure of canonical tables. However, when processing new source spreadsheet table layouts, we need to create a new set of transformation rules in CRL to get canonical tables. The rules that we used in this case are listed below.

**Rule 1** creates column labels from cells placed in a head. When there exists a cell $c_0$ in the left top corner (line 02) and a cell $c_1$ is located in the same rows and right columns (line 03) then a label is created in $c_1$ (line 05) and this label...
is associated with the category "Column Heading" (line 06) as shown in the listing below:

01 when
02 cell c0: rt == 1, cl == 1
03 cell cl: rb <= c0.rb, cl > c0.cr
04 then
05 new label cl
06 set category "Column Heading" to cl.label

**Rule 2** creates parent-child pairs from labels placed in a head. When there exists a pair of labels 11 and 12 originated from the head determined by a corner cell c0 and the cell of 11 spans the cell of 12 by columns (lines 02-05) then the parent label 11 is associated with the child label 12 (line 07) as shown in the listing below:

01 when
02 cell c0: rt == 1, cl == 1
03 label l1: cell.cl > c0.cr, cell.rb < c0.rb
04 label l2: cell.rt == l1.cell.rb + 1,
05 cell.cl >= l1.cell.cl, cell.cr <= l1.cell.cr
06 then
07 set parent l1 to l2

**Rule 3** creates row labels from cells placed in a stub. When a cell c1 is located in the same columns of the corner cell c0 (lines 02-03) then a label is created in the cell cl and in the listing below:

01 when
02 cell c0: rt == 1, cl == 1
03 cell cl: cr <= c0.cr
04 label l1: cell.rb == c0.rb,
05 cell.rb <= l1.cell.rb, cell.cl >= 1
06 then
07 set parent cl to l1

**Rule 4** creates parent-child pairs from labels placed in a head. When there exists a pair of labels 11 and 12 originated from the stub determined by a corner cell c0 and the cell of 11 spans the cell of 12 by rows (lines 02-05) then the parent label 11 is associated with the child label 12 (line 07) as shown in the listing below:

01 when
02 cell c0: rt == 1, cl == 1
03 cell cl: cr <= c0.cr
04 then
05 new label cl
06 set category "Row Heading" to cl.label

**Rule 5** creates parent-child pairs from labels placed in a stub head and labels placed in a stub. When a label 11 is originated from a cell located below the corner cell c0 and in the 1st column (lines 02-03) then the parent label originated from c0 is associated with the child label 11 (line 05) as shown in the listing below:

01 when
02 cell c0: rt == 1, cl == 1
03 label l1: cell.rt > c0.rb, cell.cl == 1
04 then
05 set parent c.label to l1

**Rule 6** splits merged cells placed in a body. When a cell c1 is located in the body determined by the corner cell c0 on several rows (lines 02-03) then it is split (line 05) as shown in the listing below:

01 when
02 cell c0: rt == 1, cl == 1
03 cell cl: rt > c.rb, cl > c0.cr, rt < rb
04 then
05 split cl

**Rule 7** creates entries from cells placed in a body. When a cell c1 is located in the body determined by the corner cell c0 (lines 02-03) then an entry is created in c1 (line 05) as shown in the listing below:

01 when
02 cell c0: rt == 1, cl == 1
03 cell cl: rt > c0.rb, cl > c0.cr
04 then
05 new entry cl

**Rule 8** associates entries with labels by the same rows and columns. When there exists a triplet of an entry e and two terminal labels 11 and 12 located in the same columns and the same rows of the cell of e respectively (lines 02-07) then both labels 11 and 12 are associated with the entry e (lines 09-10) as shown in the listing below:

01 when
02 cell c0: rt == 1, cl == 1
03 entry e
04 label l1: cell.rb == c0.rb,
05 cell.cr >= e.cell.cl, cell.cl <= e.cell.cr
06 label l2: cell.cr == c0.cr,
07 cell.rb >= e.cell.rt, cell.rt <= e.cell.rb
08 then
09 add label l1 to e
10 add label l2 to e

The second form (Form 2) is a list with only two columns. Each row puts together a heading and data, i.e. a label si and an entry di. It also merges cells for repeating headings and data. Figure 3 depicts an example of a source table Form 2 (left) and its target table (right). Form 2 does not have column headings. The ruleset expressed in CRL intended for the data canonicalization from source tables of Form 2 is listed below.

![Figure 3](https://example.com/figure3.png)

**FIGURE 3.** An example of the transformation spreadsheet data from a source table of Form 2 (left) to its target table in the canonical form (right).

**Rule 1** creates labels from cells placed in a left column. When a cell c is located in the 1st column (line 02) then a label is created in c (line 04) as shown in the listing below:
Rule 2 creates entries from cells placed in a right column and associates them with labels originated from the same rows. When a cell \( c \) is located in the 2nd column (line 02) and a label \( l \) is originated from a cell in the same row of \( c \) (line 03) then an entry is created in \( c \) (line 05) and the label \( l \) is associated with this entry (line 06) as shown in the listing below:

```
01 when
02 cell c: cl == 1
03 then
04 new label c
```

The last case 5 is originated from source tables of Form 2. We assume that a record contains only one path with only one row label, as shown in Figure 7. In all cases 1-5, each record contains only one entry.

We define a transformation algorithm for processing paths of row labels RL as follows:

```
01 Create new RL class
02 New RL class name = first RL label
03 If count of RL labels == 1 then
04 Create new property of new RL class
05 New property name = "Name"
06 If count of RL labels == 2 then
07 Create new property of new RL class
08 New property name = second RL label
```

We also define a transformation algorithm for processing paths of column labels CL as follows:

```
01 If count of CL labels == 1 then
02 Create new property of new CL class
03 // created on the step 1 of
04 // the row heading analysis algorithm
05 New property name = first CL label
06 If count of CL labels == 2 then
07 Create new CL class
08 New CL class name = first CL label
09 Create new relationship
10 New relationship name = "has a"
11 + first CL label
12 New relationship right entity = New RL class
13 // created on the step 1 of
14 // the row heading analysis algorithm
```

B. FORMALIZATION AND REPRESENTATION OF KNOWLEDGE

A canonicalized table is transformed into one fragment of knowledge formalized and represented as a conceptual model that describes a limited subset of domain concepts and relationships. Its paths of labels are interpreted as some hierarchy domain concepts (classes or attributes). This process is driven by a set of transformation rules taking into account the five cases of canonicalized table records described in [35]. Cases 1-4 originate from source tables of Form 1 and are described below.

In case 1, each of both paths in a record (a row heading and a column heading) has only one label as shown in Figure 4 (UML is used just for visualization). In case 2, a record contains a path of two row labels and a path of only one column label as also shown in Figure 4. In the latter case, labels originated from a stub of a source table can be read as hierarchical (parent-child) relationships.

```
\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure4.png}
\caption{Generating conceptual models fragments from a canonicalized table (Cases 1-2).}
\end{figure}
```

In case 3, each of both paths in a record contains a pair of labels (Fig. 5). The paths of labels can be read as hierarchical relationships.

In case 4, a path of row labels in a record has only one label, while a path of column labels in the record contains two or more labels (Fig. 6). In this case, labels originated from a head of a source table can be read as multiple parent-child relationships.

```
\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure5.png}
\caption{Generating conceptual models fragments from a canonicalized table (Case 3).}
\end{figure}
```

```
\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure6.png}
\caption{Generating conceptual models fragments from a canonicalized table (Case 4).}
\end{figure}
```

```
\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure7.png}
\caption{Generating conceptual models fragments from a canonicalized table (Case 5).}
\end{figure}
```
C. SYNTHESIZING RULE-BASED KNOWLEDGE BASES

The final phase generates source codes of a target knowledge base from a conceptual model obtained at the previous phase. The synthesis algorithm is based on the generalized method for transforming a conceptual model to a knowledge base presented in [37]. This method implies the application of metamodels and a domain-specific declarative language for describing correspondences between source elements of a conceptual model and target elements of a rule-based model. The algorithm includes the following main steps:

**Step 1:** Serialize a complete conceptual model represented in a UML class diagram to the XML format using the OMG XMI standard.

**Step 2:** Extract constructions of a rule-based model from the structure of a UML class diagram serialized in XML.

**Step 3:** Modify the obtained rule-based model by using Rule Visual Modeling Language (RVML) [37] is a UML extension designed for rule-based engineering.

**Step 4:** Generate source codes of the target knowledge base by transforming the rule-based model.

Note that the rule-based model we used can be considered as a universal tool for the intermediate representation of extracted knowledge in the form of logical rules. It does not depend on a certain knowledge representation language. This algorithm is discussed in detail in [37]. Our software provides source code generation for the following languages: CLIPS, DROOLS, and PHP.

V. IMPLEMENTATION

The approach was implemented by integrating two tools: TABBYXL [10] that extracts relational data from source spreadsheet tables, and PKBD [11] that generates and aggregates conceptual models from canonicalized tables.

TABBYXL enables software development for rule-driven data extraction from arbitrary spreadsheet tables. Such software converts tabular data to a canonical form. Both a source (arbitrary) form and a target (canonical) form are determined by user-defined rules. The two scenarios for the implementation of these rules are described below.

**Scenario 1:** The transformation rules are expressed in a domain-specific language CRL. They automatically can be translated to the Java source code of an executable application.

**Scenario 2:** The transformation rules are expressed in a general-purpose rule-based language (e.g. DROOLS or JESS). They can be executed by an appropriate rule engine that is compatible with “Java Rule Engine API”.

PKBD provides prototyping knowledge bases by using logical rules and visual modeling based on the RVML notation. It supports various data sources including mind maps, class diagrams, and spreadsheet tables. To integrate both tools, we developed a plug-in module for PKBD that realizes rules for transforming spreadsheet cell values and their relationships to the resulting domain entities (taxonomical fragments). The module also aggregates the...
fragments into a complete domain model by using the rule-based operations for clarifying entity names, merging, and separation.

VI. CASE STUDY

The developed method and tools are used when solving tasks in the field of Industrial Safety Inspection (ISI) [38]. Let’s consider our case study and evaluation in detail.

A. INDUSTRIAL SAFETY INSPECTION

ISI is a procedure required to confirm the compliance of the technical equipment state with industrial safety requirements. There are national standards and normative acts for regulating this procedure [38-40]. They define the composition and stages of ISI in a general form. In most cases, implementation details depend on the technical abilities and experience of inspecting organizations [41]. Our case study relies on the experience of the Irkutsk Research and Design Institute of Chemical and Petrochemical Engineering (IrkutskNIIhimmash) to demonstrate some details of the ISI procedure. This organization accumulated a large amount of information on technical state evaluation and risk assessment in the form of printed and electronic documents. Note, they mainly use word processors and spreadsheets without involving specialized software to prepare their ISI reports. However, representation of this information in the form of semantic structures such as conceptual models and knowledge bases can improve the efficiency of the technical condition evaluation and residual life and risk assessment. A substantial part of this information is represented as tables in ISI reports. They are of particular interest for knowledge-based engineering due to their high degree of structurization and formalization.

We design an experiment involving domain experts in ISI to demonstrate the usefulness of our approach. The considered area of the safety management of industrial equipment includes the tasks of monitoring, diagnosing, and forecasting technical conditions and risk assessment. Only the complete and adequate formalized and represented knowledge in the form of domain models provide the proper solution to these tasks.

B. SEMI-AUTOMATED FORMALIZATION AND REPRESENTATION ENGINEERING KNOWLEDGE FOR INDUSTRIAL SAFETY INSPECTION TASKS

1) OBTAINING CANONICALIZED TABLES

We used a dataset of 216 spreadsheet tables extracted from 6 ISI reports, 173 of them have a unique layout and content in 5817 cells. We selected 161 tables and transformed them to the canonical form by executing the transformation rules described above. These tables accompanied by the transformation rules are referred to as an ISI-161 dataset (ISI-161)².

Figure 8 shows some examples of these tables containing information about elements of the inspected object and results of a hardness measurement. Figure 9 depicts the corresponding canonicalized tables obtained from the source ones via TABBYXL.

2) FORMALIZATION AND REPRESENTATION OF KNOWLEDGE

The canonicalized tables of ISI-161 were transformed with the aid of PKBD. A total of 429 entities, including 59 classes (concepts), 338 attributes (properties), and 32 associations (relationships), were allocated depending on the activated aggregation rules. Note that the experts estimated that only about 56% of 429 entities were useful for further processing. The aggregation of the taxonomical fragments into a complete conceptual model reduced them to 242 entities, including 25 classes, 196 attributes, and 21 associations.

The resulting fragments were verified by the domain experts and compared with the existing ISI-models dataset provided by the IrkutskNIIhimmash (ISI models)³. In the context of comparing models, the domain experts found that 17% (69 out of 400) of concepts from ISI-models dataset

² [dataset] A.Yu. Yurin, A.O. Shigarov, N.O. Dorodnykh, V.V. Khristyuk, ISI-161: Spreadsheet tables, Mendeley Data, v1, 2019. http://dx.doi.org/10.17632/8zdymg4y96.1.
³ [dataset] A.Yu. Yurin, N.O. Dorodnykh, O.A. Nikolaychuk, A.F. Berman, A.F. Pavlov, ISI models, Mendeley Data, v1, 2019. http://dx.doi.org/10.17632/f9h2r766dk.1.
have concepts identical to the concepts from the models obtained as a result of the analysis of the ISI-161 dataset, including entities (Table I), properties and relationships (Table II). Figure 10 presents a fragment from the ISI-model dataset (01.mdl file) and the corresponding concepts from the results of the transformation.

The model validation revealed the need to use additional information sources, for example, conceptual models for describing the dynamics of technical states (ISI models) to build useful knowledge bases, in particular, for solving diagnostic and forecasting issues. These models were obtained from a survey of experts in the IrkutskNiiHimmash.

**TABLE I**

EXAMPLES OF MATCHING ENTITIES FROM DATASETS ISI-161 AND ISI-MODELS

| ISI-161       | ISI-MODELS                        |
|---------------|-----------------------------------|
| material      | material                          |
| inspection method | inspection result                |
| organization  | expert organization; enterprise   |
| parameter     | parameters; operation parameters  |
| dimensions    | geometric parameters              |
| welded joint  | information about welding         |
| hardness      | the results of hardness measurement|
| thickness     | thickness measurements            |
| point (place) of measurement | the results of thickness measurements at points |
| fitting       | fittings; flanges; information about fittings and flanges for MID |
| expert        | experts and specialists           |
| element       | elements of a technical object; technical object |

Moreover, the coincidence reaches 24% (106 of 400) when we complement the concepts from the ISI-model dataset with the relevant properties of the corresponding concepts from the ISI-161 dataset. Table III shows the quantitative characteristics of the compared datasets.

As a result, the processing of 161 spreadsheets from 6 ISI reports provided 24% of elements of the ISI domain model. This confirmed the applicability of the approach to the conceptual model construction.

**TABLE III**

QUANTITATIVE CHARACTERISTICS OF DATASETS

| DATASET / CRITERIA | ELEMENTS | CONCEPTS | PROP. | REL. |
|--------------------|----------|----------|-------|------|
| ISI-161 initial    | 429      | 59       | 338   | 32   |
| ISI-161 revised    | 242      | 25       | 196   | 21   |
| (56% of ISI-161 initial) | 400      | 98       | 249   | 53   |
| ISI-models (fragment of 21 models) | 69      | 14       | 51    | 4    |
| Corresponded (17% of ISI-models) | 106      | 14       | 88    | 4    |

It should be noted that the resulting structures were used as prototypes or drafts for ontology and knowledge bases, in particular for the software supporting ISI [41]. For this purpose, we used the feature of our software to synthesize syntactically correct source codes that include descriptions of template facts and rules for PHP, DROOLS, and CLIPS (Figure 12).

**C. EXPERIMENTAL EVALUATION**

TabbyXL and PKBD implement (a) the transformation from arbitrary tables to canonicalized ones, (b) the transformation from canonicalized tables to conceptual models, and (c) the transformation from conceptual models to source codes. To evaluate the performance of our implementation, we use the well-known measures: recall and precision. For the first transformation, they were adopted as follows:

\[
\text{recall} = \frac{|R \cap S|}{|R|}, \quad \text{precision} = \frac{|R \cap S|}{|S|},
\]

where \(R\) is a set of entities in the resulting table, and \(S\) is a set of cell values in the corresponding source spreadsheet.

For the second transformation, they are calculated as follows:

\[
\text{recall} = \frac{TP}{TP + FN}, \quad \text{precision} = \frac{TP}{TP + FP},
\]

where \(TP\) is a set of correctly transformed entities (cell values), \(FP\) is a set of incorrectly transformed entities, and \(FN\) is a set of untransformed entities that could be transformed. Table IV enumerates the results of the evaluation for transformations.

**VII. DISCUSSION**

The performance evaluation showed that using this approach we managed to transform most of the arbitrary and canonized tables. Note that the transformations were evaluated only formally. However, in the context of our case study, the ISI reports tables were not quite suitable (estimated as 17% and 24%) from a viewpoint of qualitative content evaluation of the utility of the obtained models.
An analysis of the concept features obtained from the ISI-161 and ISI-models datasets showed a difference in the emphasis of these models. While the ISI-161 dataset contains information mainly about the results of technical diagnostics of equipment, the ISI-models dataset is focused on the description of the entire ISI procedure, including tasks such as the development of an ISI program, the analysis and interpretation of the results of the diagnostic, as well as making decisions for the repair and forming a conclusion (report) for ISI. For this reason, 173 concepts from ISI-161 were not further used.

We define the main reasons that lead to reducing the recall and precision for both transformation stages. The failed cases of the first stage (the table canonicalization) were mainly caused by the following reasons: (I) new layout of source tables that is not supported by the transformation rules we developed, (II) erroneous assignment of a certain cell type for the resulting data (e.g. the date data type is assigned to a cell with the numeric data type). The errors of the second stage (tables-to-conceptual models) occurred mostly due to the following reasons: (I) imperfection of the transformation rules for processing an embedded hierarchy of concepts, in particular, skipping the third hierarchy level of concepts for row headings, (II) imperfection of the aggregation strategies.
for conceptual model fragments (e.g. the merging of “gt_20, MPa” and “gb_20, MPa” concepts that are syntactically similar but different semantically), (III) invalid cell type in the input spreadsheet data.

We made a qualitative comparison of our approach in terms of generating conceptual models from spreadsheets (Table V). Each of the considered methods fails to process the table layouts from ISI reports. Therefore, the use of our approach and software (TABBYXL and PKBD) is promising in this aspect, in particular, helps to generate higher-level abstractions (e.g., specification and a source code of rule-based knowledge bases) based on the formalized and represented knowledge.

PKBD synthesizes syntactically correct source codes based on extracted knowledge, while the internal representation of rules (ensuring its independence from a specific programming language) provides generation of fairly simple rules that do not support such specific elements as variables, calculated expressions, functions, etc. However, such structures can be added after the synthesis of the codes in their debugging and integration.

VIII. CONCLUSION

We propose a novel three-phase approach based on the spreadsheet data and conceptual model extraction and transformation to automate the formalization and representation of complex engineering knowledge in the context of the construction of knowledge bases. Our approach was implemented by the data-level integration of the two tools: TABBYXL and PKBD. TABBYXL is used to extract spreadsheet data from arbitrary spreadsheets and to transform them into the canonical form. PKBD generates a

### TABLE V

| APPROACHES / COMPARISON CRITERIA | THERINO ET AL. [9] (TANGO) | HUNG ET AL. [8] (TRAN) | HERMANS ET AL. [27] (GYRO) | CUNHA ET AL. [21] (HAEXCEL) | AMALFITANO ET AL. [20] | TABBYXL + PKBD |
|----------------------------------|-----------------------------|------------------------|-----------------------------|-----------------------------|------------------------|----------------|
| Data source                      | Web                         | Excel                  | Excel                       | Excel                       | Excel                  | Excel          |
| Source table format              | HTML                        | Excel                  | Excel                       | Excel                       | Excel                  | Excel (TabbyXL canonical tables) |
| Target format                    | OWL ontology                | XML data model         | Class                       | UML class diagram + OCL     | UML class diagram      | UML class diagram, concept map, CLIPS (depends on PKBD modules) |
| Constraints for spreadsheet layouts | Yes                         | Partial (use of XSLT/XQuery) | Yes (use of a patterns library) | Yes (use of ClassSheets) | Not defined | No (the use of CRL) |

FIGURE 8. Examples of synthesized source codes for a) PHP; b) DROOLS; c) CLIPS.
conceptual model from the canonicalized tables and synthesizes a rule-based knowledge base.

We used the ISI-161 dataset to conduct the experimental evaluation of our approach. The experiments revealed that it is suitable for processing spreadsheet data from ISI reports and generating domain models and a source code of knowledge base in the ISI area. However, there is a need to improve a set of rules for aggregating fragments of conceptual models. For example, semantic similarity can be used to improve merging classes and individual attributes. It should also be noted that the thematic connectivity of source spreadsheet tables is important for obtaining useful domain models.

The novelty of our approach is justified as follows: (I) an original canonical form of tables, providing an intermediate representation and automated processing arbitrary tables with different layouts, (II) the use of CRL, a domain-specific language for expressing rules for the transformation of spreadsheet tables from ISI reports, (III) the algorithms for converting canonicalized tables to fragments and aggregating them to a conceptual model, (IV) the use of our tools (TABBYXL and PKBD), (V) ISI-161, the new dataset of source tables and conceptual models for the ISI procedure.

Our approach is considered solely in the context of semi-automated formalization and representation of knowledge, while only spreadsheet tables are used as source data, without additional information. The pro-posed solution can support the development of knowledge-based systems in different domains.

In the future, we plan to enhance the accuracy by clarifying and involving layout properties of source tables, improving rules for aggregating and converting canonicalized tables. Table transformations can also be improved by using external taxonomies (ontologies) and annotating techniques. It is also interesting to find out whether our approach can be applied in other domains, not only ISI. However, it should be noted that a meaningful rather than formal evaluation of the results requires the simultaneous existence of both tabular data and conceptual models for a certain domain, which is quite rare.

ACKNOWLEDGMENT
This work was supported by the Russian Science Foundation, Grant No. 18-71-10001.

REFERENCES
[1] W.P. Wagner, “Trends in expert system development: A longitudinal content analysis of over thirty years of expert system case studies,” Expert Systems with Applications, vol. 76, pp. 85–96, 2017, DOI: 10.1016/j.eswa.2017.01.028.
[2] W.J.C. Verhagen, P. Berrmill-Garcia, R.E.C. van Dijk, and R. Curran, “A critical review of Knowledge-Based Engineering: An identification of research challenges,” Advanced Engineering Informatics, vol.26, pp. 5–15, 2012, DOI: 10.1016/j.aei.2011.06.004.
[3] W. Wang, M. Yang, and P.H. Seong, “Development of a rule-based diagnostic platform on an object-oriented expert system shell, Annals of Nuclear Energy,” vol.88, pp. 252–264, 2016, DOI: 10.1016/j.anucene.2015.11.008.
[4] A.F. Berman, G.S. Maltugueva, and A.Yu. Yurin, “Application of case-based reasoning and multi-criteria decision-making methods for material selection in petrochemistry,” Proceedings of the Institution of Mechanical Engineers, Part L: Journal of Materials: Design and Applications, vol. 232, no.3, pp.204-212, 2018, DOI: 10.1177/1464420715620019.
[5] F. Wang, H. Li, C. Dong, and L. Ding, “Knowledge representation using non-parametric Bayesian networks for funneling risk analysis,” Reliability Engineering & System Safety, vol.191, 106529, 2019, DOI: 10.1016/j.ress.2019.106529.
[6] L. Ramos, “Semantic Web for manufacturing, trends and open issues: Toward a state of the art,” Computers & Industrial Engineering, vol. 90, pp.444–460, 2015, DOI: 10.1016/j.cie.2015.10.013.
[7] O. Lehmberg, D. Ritzé, R. Meusel, and C. Bizer, “A large public corpus of web tables containing time and context metadata,” Proc. of the 25th International Conference Companion on World Wide Web, 2016, 75–76, DOI: 10.1145/2872518.2889386.
[8] V. Hung, B. Benatallah, and R. Saint-Paul, “Spreadsheet-based complex data transformation,” in Proc. 20th ACM Int. Conf. Inf. and Know. Management, Glasgow Scotland, UK, 2011, pp. 1749–1754, DOI: 10.1145/2056376.2056392.
[9] Y. A. Tijerino, D.W. Embley, D.W. Lonsdale, Y. Ding, and G. Nagy, “Towards Ontology Generation from Tables,” World Wide Web: Internet and Web Information Systems, vol. 8, no. 8, pp. 261–285, Sept. 2005, DOI: 10.1007/s11280-005-0360-8.
[10] A. Shigarov, V. Khristyuk, and A. Mikhailov, “TabByXL: Software platform for rule-based spreadsheet data extraction and transformation,” SoftwareX, vol. 10, p. 100270, Dec. 2019, DOI: 10.1016/j.softx.2019.100270, [Online].
[11] A.Yu. Yurin, and N.O. Dorodyukh, “Personal knowledge base designer: Software for expert systems prototyping,” SoftwareX, vol. 11, p. 100411, Jan. 2020, DOI: 10.1016/j.softx.2020.100411, [Online].
[12] M. Fiorelli, T. Lorenzetti, M.T. Piazena, A. Stellato, and A. Turbati, “Sheet2RDF: A Flexible and Dynamic Spreadsheet Import&Lifting Framework for RDF,” Proc. of the 28th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 131-140, 2015, DOI: 10.1007/978-3-319-19066-2_13.
[13] L. Han, T. Finin, C. Parr, J. Sachs, and A. Joshi, “RDF2F3: From Spreadsheets to RDF,” Proc. of the 7th International Semantic Web Conference (ISWC), pp.451-466, 2008, DOI: 10.1007/978-3-540-88564-1_29.
[14] A. Langegger, W. Woß, “XLWrap - Querying and Integrating Arbitrary Spreadsheets with SPARQL,” Proc. of the 8th International Semantic Web Conference (ISWC), pp.359-374, 2009, DOI: 10.1007/978-3-642-04930-9_23.
[15] M.J. O’Connor, C. Halaschek-Wiener, and M.A. Musen, “Mapping Master: A Flexible Approach for Mapping Spreadsheets to OWL,” Proc. of the 9th International Semantic Web Conference (ISWC), pp.194-208, 2010, DOI: 10.1007/978-3-642-17749-1_13.
[16] F. Scharfie, G. AtenZege, R. Troncy, F. Gandon, S. Villata, B. Boulier, F. Handli, L. Bihanic, G. Képeklian, F. Cotton, J. Euzenat, Z. Fan, P.Y. Vandenbusche, and D. Vanant, “Enabling Linked Data publication with the Datafit platform,” Proc. of the AAAI workshop on semantic cities. In 26th Conference on Artificial Intelligence, W10: Semantic Cities, pp. 25-30, 2012.
[17] X. Zhang, R. Di, and X. Feng, “Ontology Based Data Conversion from Spreadsheet to OWL,” Proc. of the 2012 Seventh China Grid Annual Conference, pp. 76–79, 2012, DOI: 10.1011/ACCESS.2012.17.
[18] Q. Zou, Y. Chen, W. Chu, and X. Lu, “Mining Association Rules from Tabular Data Guided by Maximal Frequent Itemsets,” In: Chu W., Young Lin T. (eds.) Foundations and Advances in Data Mining, Studies in Fuzziness and Soft Computing, vol. 180, pp. 163-181, 2009, DOI: 10.1007/978-3-642-17749-1_7.
[19] R. Alberts, and E. Francioni, “An Integrated Method Using Conceptual Modelling to Generate an Ontology-based Query Mechanism,” Proc. of the OWL: Experiences and Directions Workshop 2012 (OWLED), 849, 2012.
[20] D. Amalfitano, A. Fasolino, P. Tramontana, V.D. Simone, M. D. Mare, and S. Scala, “Information Extraction from Legacy Spreadsheet-based Information System - An Experience in the
Automotive Context,” in Proc. 3rd International Conference on Data Management Technologies and Applications (KomiIS-2014), Vienna, Austria, 2014, pp. 389–398, DOI: 10.5220/0005139603890398.

[21] J. Cunha, M. Erwig, M. Mendes, and J. Saraiva, “Model inference for spreadsheets,” Automated Software Engineering, vol. 23, no. 3, pp. 361–392, Sept. 2016, DOI: 10.1007/s10515-014-0167-x.

[22] J. Zedlitz, and N. Luttenberger, “Conceptual Modelling in UML and OWL-2,” International Journal on Advances in Software, vol.7, no.1, pp.182–196, 2014.

[23] R.R. Starr, and J.M. Parente de Oliveira, “Concept maps as the first step in an ontology construction method,” Information Systems, vol. 38, pp.771–783, 2013, DOI: 10.1016/j.is.2012.05.010.

[24] M. Mehrolhassani, and A. Elci, “Developing Ontology Based Applications of Semantic Web Using UML to OWL Conversion,” Proc. of the Open Knowledge Society, A Computer Science and Information Systems Manifesto (WSKS 2008), Communications in Computer and Information Science, vol.19, pp. 566–577, 2008, DOI: 10.1007/978-3-540-87783-7_72.

[25] E. Reynaeres, M.L. Callusco, and M.R. Galli, “A set of ontology design patterns for reengineering SBVR statements into OWL-SWRL ontologies,” Expert Systems with Applications, vol. 42, no.5, pp.2680–2690, 2015, DOI: 10.1016/j.eswa.2014.11.012.

[26] Z. Xu, Y. Ni, W. He, L. Lin, and Q. Yan, “Automatic extraction of OWL ontologies from UML class diagrams: a semantics-preserving approach,” World Wide Web, vol. 15, no. 5, pp. 517–545, 2012, DOI: 10.1007/s11280-011-0147-z.

[27] F. Hermans, M. Pinzger, and A. V. Deursen, “Automatically extracting class diagrams from spreadsheets,” LNCS: ECOOP 2010 – Object-Oriented Programming, 6183, pp. 52–75, June 2010, DOI: 10.1007/978-3-642-14107-2_4.

[28] F.S. Parreiras, and S. Staab, “Using ontologies with UML class-based modeling: The TwoUse approach,” Data & Knowledge Engineering, vol. 69, no. 11, pp. 1194-1207, 2010, DOI: 10.1016/j.datak.2010.07.009.

[29] S. Bowers, J.S. Madin, and M.P. Schildhauer, “Owllifier: Creating OWL-DL ontologies from simple spreadsheet-based knowledge descriptions,” Ecological Informatics, vol. 5, no.1, pp.19–25, 2010, DOI: 10.1016/j.ecoinf.2009.08.010.

[30] E.F. Codd, “A Relational Model of Data for Large Shared Data Banks,” Commun. ACM 13, vol. 6, pp. 377–387, June 1970, DOI: 10.1145/362384.362685.

[31] E. Seidewitz, “What models mean,” IEEE Software, vol. 20, no. 5, pp. 26–32, Oct. 2003, DOI: 10.1109/MS.2003.1231147.

[32] M. Fowler, “UML Distilled: A Brief Guide to the Standard Object Modeling Language,” 3 ed., USA: Addison-Wesley Professional, 2018.

[33] I. Davies, P. Green, M. Rosemann, M. Indulska, and S. Gallo, “How do practitioners use conceptual modeling in practice?,” Data & Knowledge Engineering, vol.58, no. 3, pp. 358–380, Sep. 2006, DOI: 10.1016/j.datak.2005.07.007.

[34] A. Shigarov, and A. Mikhailov, “Rule-based spreadsheet data transformation from arbitrary to relational tables,” Information Systems, vol. 71, pp. 123–136, Nov. 2017, DOI: 10.1016/j.is.2017.08.004.

[35] N.O. Dorodnykh, A.Yu. Yurin, and A.O. Shigarov, “Conceptual Model Engineering for Industrial Safety Inspection Based on Spreadsheet Data Analysis,” Communications in Computer and Information Science. Modelling and Development of Intelligent Systems (MDIS 2019), Jan. 2020, pp. 51–65. DOI: 10.1007/978-3-030-39237-6_4.

[36] V.I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” Soviet Physics Doklady, USSR, Tech. Rep. 8, 1966.

[37] A.Yu. Yurin, N.O. Dorodnykh, O.A. Nikolaychuk, and M.A. Grishenko, “Prototyping Rule-Based Expert Systems with the Aid of Model Transformations,” Journal of Computer Science, vol. 14, no. 5, pp. 680–698, Apr. 2018, DOI: 10.3844/jcssp.2018.680.698.

[38] National Standard of Russian Federation 12.0.010-2009, Occupational safety standards system, Occupational safety and health management systems, Hazard and risks identification and estimation of risks, Moscow, Russia, 2011.

[39] API 580, Recommended Practice for Risk Based Inspection, 1st ed., American Petroleum Institute, Washington D.C., USA, 2009.

[40] M. Bertolini, M. Bevilacqua, F.E. Ciarpica, and G. Giacchetta, “Development of risk-based inspection and maintenance procedures for an oil refinery,” J. Loss Prev. Process Ind., vol. 22, pp. 244–253, Mar. 2009, DOI: 10.1016/j.jlp.2009.01.003.

[41] A.F. Berman, O.A. Nikolaiuch, A.Yu. Yurin, and K. A. Kuznetsov, “Support of Decision-Making Based on a Production Approach in the Performance of an Industrial Safety Review,” Chemical and Petroleum Engineering, vol. 50, no. 11-12, pp. 730–738, Mar. 2015, DOI: 10.1007/s10556-015-9970-x.