Towards Ontology Engineering Based on Transformation of Conceptual Models and Spreadsheet Data: A Case Study

Nikita O. Dorodnykh[1] and Aleksandr Yu. Yurin[1,2(✉)]

[1] Matrosov Institute for System Dynamics and Control Theory,
Siberian Branch of the Russian Academy of Sciences,
134, Lermontov Street, Irkutsk 664033, Russia
iskander@icc.ru
[2] Irkutsk National Research Technical University,
83, Lermontov Street, Irkutsk 664074, Russia

Abstract. The ontology engineering is a complex and time-consuming process. In this regard, methods for automated formation of ontologies based on various information sources (e.g., databases, spreadsheets data, and text documents, etc.) are being actively developed. This paper presents a case study for the domain ontology engineering based on analysis and transformation of conceptual models and spreadsheet data. The analysis of conceptual models, which are serialized using XML, provides the opportunity to develop content ontology design patterns. The specific concepts for filling obtained ontology design patterns are resulted from the transformation of spreadsheet data in the CSV format. In this paper, we present statement of the problem and the approach for its solution. The illustrative example describes ontology engineering for the industrial safety inspection tasks.

Keywords: Ontology engineering · Ontology design patterns · OWL · Conceptual models · Spreadsheets · Transformations · Industrial safety inspection

1 Introduction

Ontologies [1] gained a wide popularity among knowledge base developers as powerful means for representing knowledge in the form of semantic nets with support for the description logic by the standard of the W3C consortium – OWL (Web Ontology Language). Ontology engineering is a complex and time consuming process. Generally, ontology engineering is performed in specialized ontological editors (e.g., Protégé, OntoStudio, WebOnto, Fluent Editor, etc.). However, they mainly target highly qualified specialists (programmers) and restrict access to any other sources of data and knowledge for ontology formation.

In this regard, methods for automated formation of ontologies based on various information sources (e.g., databases, spreadsheets data, and text documents, etc.) are being actively developed. One of such sources is conceptual models of subject domains (e.g., diagrams, schemes, flowcharting, concept maps, mind maps, etc.). Conceptual

modeling tools are more attractive for domain experts. Such tools make it possible to manipulate domain entities including domain-specific notations, for example, event trees, fault trees, Ishikawa diagrams, mind maps and others. The analysis of conceptual modeling tools including IHMC CmapTools, FreeMind, Coggle, Mindjet MindManager, TheBrain, XMind, IBM Rational Rose, Star UML revealed two main issues: the lack of ability of source code synthesis for ontologies or knowledge bases, but also the absence of a general document standard for representation of concepts and relationships. At the same time, most of such software uses incompatible XML-like formats. However, these formats can be used for automatic retrieving of information about concepts and relationships.

The use of conceptual models for ontology engineering provides reusing of a large amount of accumulated heterogeneous information and involvement of domain experts in the knowledge formalization process. Thus, the development of new methods and software for the ontology engineering based on the transformation of various conceptual domain models is relevant and confirmed by the following examples [2, 3].

It should be noted that automated analysis and transformation of conceptual models provides the ability to obtain ontology fragments at the abstract conceptual level (T-Box level).

At the same time, it is rather difficult to obtain specific knowledge (instances and their values) on the basis of conceptual models. For this purpose, it is necessary to use rather large sets of already existing previously accumulated information presented in the form of various structured data. In particular, such data can be spreadsheet data. Processing and extracting knowledge from them is also a relevant task today.

This paper describes the case study of ontology engineering based on automated analysis and transformation of conceptual models and spreadsheet data. Main results of transformation of conceptual models are ontology schemas in the form of content ontology design patterns (ODPs) presented in the OWL format. Main results of transformation of spreadsheet data are domain ontologies which are formed taking into account the ODPs obtained earlier.

Therefore, the automated ontology formation at the T-Box and A-Box levels involves a set of sequential transformations:

1. conceptual models to an ontology schema at the T-Box level;
2. arbitrary spreadsheet data to canonical form as an intermediate data representation;
3. canonical tables to ontology instances (domain ontology at the A-Box level).

Thus, such transformation can be formalized as follows:

$$T_1 : CM^{XML} \rightarrow ODP^{OWL},\ T_2 : SD^{CSV} \rightarrow CT^{CSV},$$
$$T_3(ODP^{OWL}) : CT^{CSV} \rightarrow Ont^{OWL}$$

where CM^{XML} is a source conceptual model in the XML format; ODP^{OWL} is ODP in the OWL format; SD^{CSV} is an arbitrary source spreadsheet data in the CSV format; CT^{CSV} is a target canonical (relational) table in the CSV format; Ont^{OWL} is a domain ontology with instances in the OWL format, T_1, T_2, T_3 are transformation operators which contain a set of transformation rules.

The approach proposed includes three steps and employs the following software: Knowledge Base Development System (KBDS) [4] for transformation of conceptual models to ODPs; TabbyXL [5] for extraction of canonical (relational) tables from arbitrary spreadsheet data in the CSV format; TabbyOWL prototype for OWL domain ontology generation based on transformation of canonical tables. Industrial safety inspection (ISI) tasks are selected as a subject domain.

The ISI is a procedure for assessing the technical condition of industrial facilities, in order to determine the residual life of the operated equipment and the degradation processes occurring on it.

Analysis of the ISI procedure stages [6] showed that the implementation of some stages (forming a map of initial data, developing the ISI program, analysis of diagnostic results including interpretation, etc.) requires the processing of a large volume of semi-formalized information, which can be enhanced by building and using ontologies.

2 Background

Ontology engineering for ISI tasks includes three stages of transformation, and each of them requires the use of specific methods and tools, as well as determining the type of ODPs created.

2.1 Ontology Design Pattern Engineering

Ontology Design Patterns (ODPs) [7] are a way for fixing typical solutions in ontology engineering that helps to avoid some frequently repeated mistakes. Currently, there are some ODP catalogues, e.g., Association for Ontology Design & Patterns (ODPA) [8], which contain different types of ODPs: structural, correspondence, content, reasoning, presentation, and lexico-syntactic. The content and structural ODPs are the most popular and frequently used ones.

Content ODPs encode conceptual design patterns and provide solutions to domain modeling problems. They affect only a specific region of an ontology dealing with such domain modeling problems. Logical ODPs are independent from a specific domain of interest (i.e. they are content-independent), but they depend on an expressivity of logical formalism used for representation. In other words, logical ODPs help to solve design problems where primitives of a representation language do not directly support certain logical constructs. Architectural ODPs affect the overall shape of ontology. Their goal is to constrain “how the ontology should look like”. Thus, the use of common ontology fragments in the form of structural and content ODPs allows us to simplify and speed up further refinement (modification) of ontologies obtained through the use of standard solutions.

ODPs contain a set of concepts and relationships reflecting a semantic structure of a domain at a fairly high abstract level. In some cases this is not enough for solving practical tasks. In particular, structured data sets in the form of spreadsheet data can be used to fill ODPs.

In this paper, content ODPs will be created as the first step in the complex automation of ontology engineering.

2.2 Ontology Engineering Based on Model Transformations

Today, few researchers construct ODPs on the basis of conceptual model transformations. However, there are some examples with transformation support of different conceptual models for ontology engineering. A metamodel-driven model transformation approach for interchanging rules between OWL along with Semantic Web Rule Language (SWRL) and Object Constrained Language (OCL) along with Unified Modeling Language (UML) is proposed in [9]. REWERSE Rule Markup Language (R2ML) is used as an intermediate knowledge representation (rules). ATL is used to describe the transformation rules. In [10] researchers present an approach to transforming UML class diagrams into OWL ontologies using the Query/View/Transformation (QVT) standard.

Nowadays, the world accumulates a big volume of various data presented in the XML format. This format is a universal and most common way of integrating software and providing the exchange of information between applications. There are examples of transforming XML-like data into target ontologies [11–14]. Some solutions aim at generating linked data in the form of Resource Description Framework (RDF) triplets based on the transformation of various source XML data. In particular, the W3C consortium has proposed the Gleaning Resource Descriptions from Dialects of Languages (GRDDL) standard [15] for the support of the XML to RDF conversion. Lange in [16] proposes an extensible framework for extracting RDF in various notations from various XML languages, which can easily be extended to additional input languages. The implementation is done in eXtensible Stylesheet Language Transformations (XSLT) with a command-line frontend and a Java wrapper. A new query language called XSPARQL was introduced in [17], and it combines XQuery and SPARQL. It allows querying XML and RDF data, as well as converting data from one format to another. The implementation of XSPARQL is based on rewriting the XSPARQL query as a semantically equivalent XQuery query.

In this paper, we propose to use our Knowledge Base Development System (KBDS) [18] and a declarative domain-specific language for description of model transformations called Transformation Model Representation Language (TMRL) [19]. This tool is used to create converters for most XML formats and eliminates dependency on a specific type of conceptual models and tools. The tool and the language have been tested in the analysis of UML diagrams of classes represented in the XMI format.

2.3 Spreadsheet Data Analysis

Spreadsheet data can be used to fill ODPs. Over the past two decades, the methodological approaches for extracting and transforming data from spreadsheets have been actively developed. In particular, methods for the role and structural spreadsheet analysis (restoration of relations between cells) are proposed in [20–23]. The research in this area is also focused, in particular, on specialized tools for extraction and transformation arbitrary spreadsheet data into a structured form, including tabular data transformation systems [24–26], linked data extraction systems [27, 28], transformation systems of arbitrary tables to a relational form based on the search for critical cells [23], relational data extraction systems from spreadsheets with header hierarchies [22]. The

tools considered have similar goals (to transform spreadsheet data from an arbitrary form to a relational form). However, they use specified models of source spreadsheets, in which their physical and logical composition (layout) are mixed. This fact limits the use of these tools for processing arbitrary spreadsheets presented in statistical reports.

Our review showed that the solutions considered above (each separately) fail to process the layouts found in ISI reports. The TabbyXL, which we developed in [5], looks promising in this aspect. TabbyXL is a command-line tool for spreadsheet data canonicalization. This tool is used to produce flat relational (normalized) spreadsheet tables from semi structured tabular data. The tool operates with CSV format documents. Data structures for representing table elements (cells, entries, labels, and categories). TabbyXL has the ability to customize certain layout types by using a domain-specific language – Cells Rule Language (CRL) [5].

2.4 RDF/OWL Ontology Generating from Spreadsheet Data

There are several recent studies that deal with spreadsheet data transformation. Such tools as RDF123, csv2rdf4lod, Datalift, Any2OWL, Excel2OWL, Spread2RDF, Any23, TopBraid Composer are used to solve issues of converting spreadsheet data to RDF or OWL formats. Some of the solutions also include own domain-specific languages: XLWrap, Mapping Master and RML.

Quite a few solutions are available in this area, especially for the generation of RDF documents. However, the generation of ontologies in the OWL format is usually poorly supported, which is why we develop our own OWL generator, focused on the format of canonical TabbyXL tables [5].

3 The Proposed Approach

The approach used in this study includes methodology and means for its implementation.

3.1 Methodology

The methodology based on the formal problem statement that we presented in introduction includes three main steps.

Step 1: Forming content ODPs based on the transformation of conceptual models. This step consists of the following main actions:

1.1 Forming a conceptual model. A domain expert represents their own knowledge in the form of a conceptual information model (perhaps, using domain-specific visual notations).

1.2 Serializing a conceptual model. The conceptual model is represented in the XML-based format for further transformation.

It should be noted that Actions 1.1 and 1.2 can be implemented using various software to support conceptual modeling and serializing models in the form of XML documents.

1.3 Analysing a XML document structure. This action involves extracting elements, their attributes and relationships from the XML tree.

1.4 Forming an ODP code in the OWL format. The main objective of this action is to obtain typical ontological fragments in the form of a set of classes and their relationships (object properties), which describe a certain domain and are based on the extracted XML elements.

A declarative domain-specific language called Transformation Model Representation Language (TMRL) [19] is used in Actions 1.3 and 1.4. This language is implemented in the Knowledge Base Development System (KBDS) [18] and provides a scenario (program) for model transformations. TMRL is designed for converting conceptual models into knowledge bases only.

The model transformation is one of the major concepts in the model-oriented approach (Model-Driven Engineering) [29]. It is an automatic generation of a target model from a source model according to a set of transformation rules. These rules together describe how a model in a source language can be transformed into a model in a target language. Consequently, a transformation rule is a description of how one or more constructs in a source language can be transformed into one or more constructs in a target language. At the same time, the use of meta-modeling is one of the main approaches to defining an abstract syntax of languages, including conceptual modeling and knowledge representation languages.

Thus, in 1.3 and 1.4 a user forms a transformation scenario in TMRL. This scenario is a set of rules for transformation of XML document elements of the conceptual model to ontological constructs. Each rule contains a correspondence between elements of the source and target metamodels with a certain priority (sequence) of this rule for an interpreter. The XML Schema is used as a metamodel for XML documents. The OWL language description is used as a metamodel for ODPs.

1.5 Editing ODP code. This action is additional and represents a refinement (modification) of the ODP obtained with the aid of various ontological modeling tools, for example, Protégé.

Thus, the main result of this step is a set of content ODPs, which define the ontology schema at the T-Box level.

Step 2: Transforming source spreadsheet data with an arbitrary layout to the canonical (relational) form. This step consists of the following main actions:

2.1 Analyzing the CSV file of spreadsheet data by using Cells Rule Language (CRL) rules [5]. CRL is a domain-specific language for expressing table analysis and interpretation rules. A set of the rules can be implemented for a specific task characterized by requirements for source and target data.

2.2 Canonicalization. The process of table canonicalization begins with loading tabular data from CSV files via Apache POI. Recovered semantics (entries, labels, and categories) allow transformation of the source spreadsheet data into the canonical form. The canonical form requires the topmost row to contain field (attribute) names. Each of the remaining rows is a record (tuple). It obligatorily includes the field named DATA that contains entries. Each extracted category constitutes a field that contains its labels. Each record presents recovered relationships between an entry and labels in each category.

Thus, let us formally define the canonical table for our case on the basis of [6]:

$$CT^{CSV} = \{DATA, RowHeading, ColumnHeading\},$$

where *DATA* is a data block that describes literal data values (named "entries") belonging to the same datatype (e.g., numerical, textual, etc.), *RowHeading* is a set of row labels of the category, *ColumnHeading* is a set of column labels of the category. The values in cells for heading blocks can be separated by the "|" symbol that is intended to divide categories into subcategories. Thus, the canonical table denotes hierarchical relationships between categories (headings). Detailed description of this process is presented in [5]. The result of this step is tables in the unified canonical form prepared for their further automated processing.

Step 3: Generating OWL domain ontology from canonical tables using ODPs. This step consists of the following main actions:

3.1 Semantic canonical table interpretation using obtained ODPs. This process is semi-automatic and aims to linking *RowHeading* and *ColumnHeading* cell values with entities of previously obtained ODPs (the ontology schema at the T-Box level). Each canonical table corresponds to a specific class of this ontology. In this case, a set of concept candidates (classes and data properties) for each *RowHeading* and *ColumnHeading* cell is determined basing on the obtained ontology schema. A cell value with separator ("|") is interpreted as a hierarchy of either classes ("subClassOf" construction) or data properties ("subPropertyOf" construction). Thus, ontology concepts, which are closest to each label, are determined. For this purpose, it is necessary to calculate an aggregated assessment (for example, a linear convolution or any other proximity measure) for all concept candidates and select a referent concept with the maximum rating. Note that context for canonical table and ontology schema should be equal. Otherwise, the aggregated assessments would be low due to the lack of correspondences between labels and ontology entities.

3.2 Forming a domain ontology at the A-Box level. The main objective of this action is to generate OWL axioms for representation of instances based on the obtained canonical table to supplement the existing ontology schema (the content ODP) with concrete instances. The obtained canonical table is divided into blocks (row sets) that correspond to a certain category group from *RowHeading* and *ColumnHeading*. At the same time, this group is annotated with a class from the ontology schema. Thus, each such row set is interpreted as an ontology instance and entities from *DATA* corresponding to data property values of instance.

3.2 Implementation

We implement our approach using KBDS, TabbyXL and TabbyOWL that provide transformations for Steps 1, 2, and 3, respectively.

KBDS is a tool for knowledge base engineering that performs transformation of various conceptual models using TMRL. KBDS is designed with the help of Model-View-Controller (MVC) design pattern, PHP, and Yii2 framework. jQuery and jsPlumb libraries are used to build graphic editors. KBDS has a client-server architecture and is described in detail in [18].

TabbyXL [5] is a command-line tool for spreadsheet data canonicalization. This tool is used to produce flat relational (normalized) spreadsheet data from semi structured tabular data. The tool operates with CSV format documents. Data structures for representing table elements (cells, entries, labels, and categories) are Java classes developed in accordance with naming conventions of JavaBeans 5 specification.

TabbyOWL is a tool (converter) for transformation of canonical tables into OWL ontology at the A-Box level. The input data for a converter is a set of transformation rules and a source canonical table. The output data is an ontology containing instances, which specifies the previously obtained content ODP. Thus, the main result is the domain ontology that reflects the abstract level of domain knowledge in terms of classes and their properties and contains specific axioms. These axioms can be useful for their further usage in intelligent systems for solving various types of domain issues.

3.3 Case Study

As an example, consider the ISI ontology engineering:

Step 1: Forming content ODPs based on transformation of conceptual models.

1.1 Forming conceptual models in the form of concept maps. At this step, conceptual models in the form of concept maps for ISI are used.

Conceptual models were created within the project for automation of the ISI procedure [30]. These models reflect different aspects and concepts of the ISI procedure, such as: technical object, geodesic measurements analysis, visual measurement control, nondestructive testing, hardness measurement, rapid diagnosis, technical conditions assessment, wall thickness test, technical documentation, testing, the ISI report generation and degradation processes that take place within the inspected object (e.g., hydrogen embrittlement, etc.) and are influenced by a combination of factors. The dataset of conceptual models [31] includes 26 models containing information about 235 concepts and 208 relationships. IBM Rational Rose and IHMC CmapTools were used when building models.

Figure 1 shows a fragment of a concept map describing storage tank elements.

1.2 Serializing concept maps to XTM (XML Topic Maps) format using IHMC CmapTools.

1.3 and 1.4 Analysing XTM document structures and forming the ODP code in the OWL format based on the transformation model in TMRL. This model is created by a user through a visual transformation model editor (a KBDS module) (see Fig. 2). The user defines correspondences between the source and target metamodel elements and assigns priority to each transformation rule.

Table 1 describes the obtained correspondences.

On the basis of the obtained correspondences, the transformation model is generated on TMRL. A fragment of the transformation model describing the transformation rules is given below:

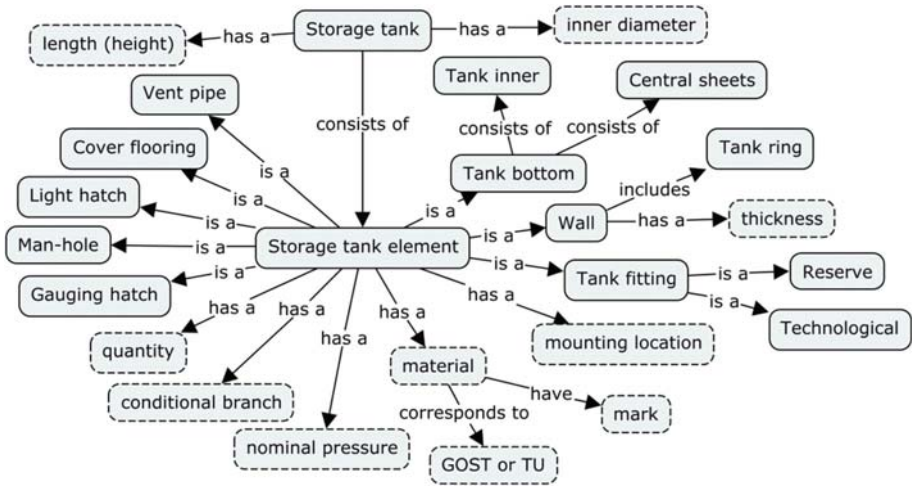

Fig. 1. A concept map fragment for a storage tank prepared with IHMC CmapTools.

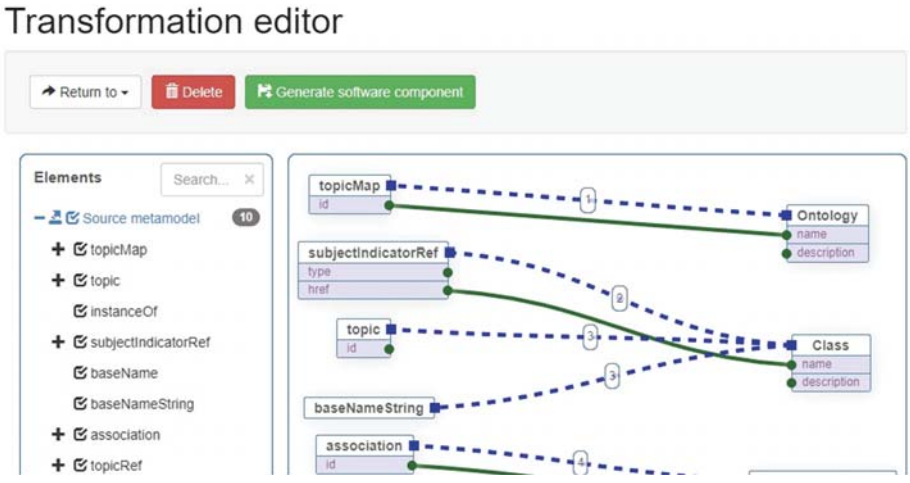

Fig. 2. GUI of the transformation model editor (a KBDS module).

Table 1. The main correspondences between XTM and OWL elements.

XTM	OWL
topicMap	Ontology
topic/subjectIndicatorRef = “concept”	Class
topic/instanceOf	subClassOf
subjectIdentity/topicRef	equivalentClass
baseName/baseNameString	rdf:about
topic/subjectIndicatorRef = “linkingPhrase”	ObjectProperty
association/instanceOf	subPropertyOf
association/subjectIdentity/topicRef	equivalentProperty
member roleSpec = “incoming”/topicRef	rdfs:domain
member roleSpec = “outcoming”/topicRef	rdfs:range
occurrence/resourceData	DatatypeProperty

```
Transformation XTM metamodel to Ontology metamodel {
   Rule topicMap to Ontology priority 1 [
      Ontology(name) is topicMap(id)
   ]
   Rule subjectIndicatorRef to Class priority 2 [
      Class(name) is subjectIndicatorRef(href)
   ]
   Rule (topic, baseNameString) to Class priority 3 [
      Class(name) is baseNameString [
         if (subjectIndicatorRef(href) == '#concept')
      ]
   ]
   Rule (topic, baseNameString) to DatatypeProperty priority 6 [
      DatatypeProperty(name) is baseNameString [
         if (subjectIndicatorRef(href) == '#linkingPhrase)
      ]
   ]
   ...
}
```

The KBDS transformation module (converter) uses this transformation model in TMRL and translates the XTM document to the OWL ODP code.

The OWL code (RDF/XML syntax) fragment of the obtained ODP is presented below:

```
<owl:Class rdf:about="StorageTank"/>
<owl:Class rdf:about="StorageTankElement">
  <rdfs:subClassOf rdf:resource="StorageTank" />
</owl:Class>
<owl:DatatypeProperty rdf:about="ConditionalBranch">
  <rdfs:domain rdf:resource="#StorageTankElement"/>
  <rdfs:range rdf:resource="xsd:positiveInteger"/>
</owl:DatatypeProperty>
```

Thus, concepts and relationships of concept maps were mapped into a set of ontological classes and their properties. The obtained ontology fragments at the T-Box level describe the patterns of various degradation processes, as well as the structure of petrochemical mechanical systems.

1.5 The OWL code obtained does not require any changes.

Step 2: Transforming spreadsheet data in the CSV format to the canonical (relational) form.

2.1 Analysis of CSV files is performed using TabbyXL. Six ISI reports were analyzed and 216 spreadsheets were extracted from these reports to fill the obtained ODPs by specific values (data) within the case study.

Figure 2(a) shows an arbitrary source table fragment that contains information about the storage tank.

2.2 The resulted canonical tables (see Fig. 2b) are generated based on analysis and transformation of the source arbitrary tables (see Fig. 2a).

The process of TabbyXL setting, layouts, discovered tables and rules for their processing are described in [32].

Step 3: Generating OWL domain ontology from the canonical tables.

3.1 and 3.2 Semantic interpretation of obtained canonical tables using generated ODPs and forming domain ontology at the A-Box level. These activities are made with the use of TabbyOWL prototype. In our case, the fragment presented in Fig. 3(b) corresponds to a single instance (individual) of "Technological" class ("TankFitting" subclass) from the resulting ontology schema at T-Box level. At the same time, labels in *RowHeading* cells describe the class, which the instance belongs. Labels in the *ColumnHeading* cells describe the data properties of this instance. Entities in *DATA* cells describe specific values for data properties of this instance.

The OWL code fragment in RDF/XML syntax of the obtained instance is presented below:

a) A source spreadsheet table

Name	Quantity, pcs.	Conditional branch, mm	Nominal pressure, MPa (kgf / cm2)	Mounting location	Material	
					Mark	GOST or TU
Inlet-outlet	*2*	*80*		*Wall*	*Steel 20*	*GOST1050*
Man-hole	*1*	*500*		*Wall*	*St3*	*GOST380*
Reserve	*1*	*200*		*Wall*		
Technological	*1*	*20*		*Wall*	*Steel 20*	*GOST1050*
Technological	*1*	*50*	*1,6 (16)*	*Wall*		
Technological	*1*	*250*		*Wall*		
Light hatch	*2*	*500*		*Roof*	*St3*	*GOST380*
Gauging hatch	*1*	*150*		*Roof*	*Steel 20*	*GOST1050*
Vent pipe	*1*	*150*		*Roof*		
Technological	*1*	*150*		*Roof*		

b) A canonicalized table

DATA	RowHeading	ColumnHeading
GOST1050	Technological	Material | GOST or TU
Steel 20	Technological	Material | Mark
Roof	Technological	Mounting location
1,6 (16)	Technological	Nominal pressure, MPa (kgf / cm2)
150	Technological	Conditional branch, mm
1	Technological	Quantity, pcs.
...		

Fig. 3. An example of a source spreadsheet data (a) and a canonical table (b) containing information about the storage tank elements.

```
<Technological rdf:about="Technological1">
  <MountingLocation rdf:datatype="xsd:string">Roof</MountingLocation>
  <Quantity rdf:datatype="xsd:positiveInteger">1</Quantity>
  <ConditionalBranch rdf:datatype="xsd:positiveInteger">150</ConditionalBranch>
  <NominalPressure rdf:datatype="xsd:string">1,6 (16)</NominalPressure>
  <Mark rdf:datatype="xsd:string">Steel 20</Mark>
  <GOSTorTU rdf:datatype="xsd:string">GOST1050</GOSTorTU>
</Technological>
```

In this example, 15 classes and 10 properties-values were obtained.

4 Discussion

In accordance with the paper objective we have made ontological engineering for ISI tasks. Methodologically, the chain of sequential transformations was made: conceptual models to ontology schema at the T-Box level; arbitrary spreadsheet data to canonical form as an intermediate data representation; canonical tables to ontology instances (domain ontology at the A-Box level).

In this case, the author's software tools were used: Knowledge Base Development System (KBDS) [18] for transformation of conceptual models to ODPs; TabbyXL [5] for extraction of canonical (relational) tables from arbitrary spreadsheet data in CSV format; the TabbyOWL prototype for the OWL domain ontology generation based on transformation of canonical tables.

The following estimates were obtained: recall – 85, precision – 89, when using KBDS for a set of 26 conceptual models containing 256 concepts and 208 relationships. The losses of recall and precision are mainly due to the complexity of transformation of the following types of correspondences for elements:

- *An ambiguous correspondence* (*synonymy*) – several XML Schema elements correspond to one of an OWL metamodel element,
- *An indistinguishable correspondence* (*homonymy*) – one of the XML Schema element corresponds to several OWL metamodel elements.

An indistinguishable correspondence can be illustrated as follows: it is necessary to map the element "*topic*" (which is a successor of the element "*concept*") into several ontological constructions: «*Class*» and «*DatatypeProperty*». To do this, a user has to configure the transformation model by adding a conditional statement to TMRL.

TabbyXL processed all tables from this data set. The TabbyXL experiment results on precision and recall for other data sets, in particular, TANGO [6] are presented in [5].

5 Conclusion

In this paper, we investigated the ability to automate the ontologies engineering on the basis of analysis and transformation of conceptual models and spreadsheet data. The methodology proposed enables reusing of a large amount of accumulated heterogeneous information: spreadsheet data and conceptual models for the automated formation of domain ontologies, which contain knowledge both at abstract (T-Box) and specific (A-Box) levels.

It should be noted that only 30% of tables from the processed ISI reports were used for filling ODPs because of the difficulties with the automatic interpretation of canonical table cell values (their semantics). This challenge requires a more detailed study of the table interpretation problem in future.

The ontologies obtained are used for prototyping rule-based expert system [33, 34].

Acknowledgement. The contribution of this work was supported by the Russian Science Foundation under Grant No. 18-71-10001.

References

1. Guarino, N.: Formal ontology in information systems. In: The First International Conference on Formal Ontology in Information Systems (FOIS 1998), vol. 46, pp. 3–15 (1998)
2. Starr, R.R., de Oliveira, J.M.P.: Concept maps as the first step in an ontology construction method. Inf. Syst. **38**, 771–783 (2013). https://doi.org/10.1109/EDOCW.2010.43
3. Herrero-Zazo, M., Segura-Bedmar, I., Martínez, P.: Conceptual models of drug-drug interactions: a summary of recent efforts. Knowl.-Based Syst. **114**, 99–107 (2016). https://doi.org/10.1016/j.knosys.2016.10.006
4. Berman, A.F., Dorodnykh, N.O., Nikolaychuk, O.A., Pavlov, N.Y., Yurin, A.Yu.: Fishbone diagrams for the development of knowledge bases. In: The 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2018), pp. 967–972 (2018). https://doi.org/10.23919/MIPRO.2018.8400177
5. Shigarov, A.O., Mikhailov, A.A.: Rule-based spreadsheet data transformation from arbitrary to relational tables. Inf. Syst. **71**, 123–136 (2017). https://doi.org/10.1016/j.is.2017.08.004
6. Tijerino, Y.A., Embley, D.W., Lonsdale, D.W., Ding, Y., Nagy, G.: Towards ontology generation from tables. World Wide Web **8**(3), 261–285 (2005). https://doi.org/10.1007/s11280-005-0360-8
7. Hitzler, P., Gangemi, A., Janowicz, K., Krisnadhi, A.A., Presutti, V.: Ontology Engineering with Ontology Design Patterns: Foundations and Applications. Studies on the Semantic Web. IOS Press/AKA, Amsterdam (2016)
8. Association for ontology design & patterns (ODPA). http://ontologydesignpatterns.org/wiki/ODPA. Accessed 18 Mar 2019
9. Milanović, M., Gašević, D., Giurca, A., Wagner, G., Devedžić, V.: Bridging concrete and abstract syntaxes in model-driven engineering: a case of rule languages. Soft. Pract. Exp. **39** (16), 1313–1346 (2009). https://doi.org/10.1002/spe.938
10. Zedlitz, J., Luttenberger, N.: Conceptual modelling in UML and OWL-2. Int. J. Adv. Softw. **7**(1), 182–196 (2014)
11. Bohring, H., Auer, S.: Mapping XML to OWL ontologies. In: Leipziger Informatik-Tage, vol. 72, pp. 147–156 (2005)
12. Rodrigues, T., Rosa, P., Cardoso, J.: Moving from syntactic to semantic organizations using JXML2OWL. Comput. Ind. **59**(8), 808–819 (2008). https://doi.org/10.1016/j.compind.2008.06.002
13. O'Connor, M.J., Das, A.K.: Acquiring OWL ontologies from XML documents. In: The 6th International Conference on Knowledge Capture (K-CAP 2011), pp. 17–24 (2011). https://doi.org/10.1145/1999676.1999681
14. Bedini, I., Matheus, C., Patel-Schneider, P.F., Boran, A., Nguyen, B.: Transforming XML schema to OWL using patterns. In: The 2011 IEEE Fifth International Conference on Semantic Computing, pp. 102–109 (2011). https://doi.org/10.1109/ICSC.2011.77
15. Gleaning resource descriptions from dialects of languages (GRDDL). https://www.w3.org/TR/grddl/. Accessed 18 Mar 2019
16. Lange, C.: Krextor - an extensible framework for contributing content math to the web of data. In: International Conference on Intelligent Computer Mathematics, pp. 304–306 (2011). https://doi.org/10.1007/978-3-642-22673-1_29
17. Bischof, S., Decker, S., Krennwallner, T., Lopes, N., Polleres, A.: Mapping between RDF and XML with XSPARQL. J. Data Semant. **1**(3), 147–185 (2012). https://doi.org/10.1007/s13740-012-0008-7

18. Dorodnykh, N.O.: Web-based software for automating development of knowledge bases on the basis of transformation of conceptual models. Open Semant. Technol. Intell. Syst. **1**, 145–150 (2017)
19. Dorodnykh, N.O., Yurin, A.Yu.: A domain-specific language for transformation models. In: CEUR Workshop Proceedings. Information Technologies: Algorithms, Models, Systems (ITAMS 2018), vol. 2221, pp. 70–75 (2018)
20. Mauro, N., Esposito, F., Ferilli, S.: Finding critical cells in web tables with SRL: trying to uncover the devil's tease. In: Proceedings of the 12th International Conference on Document Analysis and Recognition, pp. 882–886 (2013). https://doi.org/10.1109/ICDAR.2013.180
21. Adelfio, M., Samet, H.: Schema extraction for tabular data on the web. Proc. VLDB Endow. **6**(6), 421–432 (2013). https://doi.org/10.14778/2536336.2536343
22. Chen, Z., Cafarella, M.: Integrating spreadsheet data via accurate and low-effort extraction. In: Proceedings of the 20th ACM SIGKDD International Conference Knowledge Discovery and Data Mining, pp. 1126–1135 (2014). https://doi.org/10.1145/2623330.2623617
23. Embley, D.W., Krishnamoorthy, M.S., Nagy, G., Seth, S.: Converting heterogeneous statistical tables on the web to searchable databases. Int. J. Doc. Anal. Recogn. **19**(2), 119–138 (2016). https://doi.org/10.1007/s10032-016-0259-1
24. Kandel, S., Paepcke, A., Hellerstein, J., Heer, J.: Wrangler: interactive visual specification of data transformation scripts. In: Proceedings of the SIGCHI Conference Human Factors in Computing Systems, pp. 3363–3372 (2011). https://doi.org/10.1145/1978942.1979444
25. Hung, V., Benatallah, B., Saint-Paul, R.: Spreadsheet-based complex data transformation. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 1749–1754 (2011). https://doi.org/10.1145/2063576.2063829
26. Harris, W., Gulwani, S.: Spreadsheet table transformations from examples. ACM SIGPLAN Not. **46**(6), 317–328 (2011). https://doi.org/10.1145/1993316.1993536
27. O'Connor, M.J., Halaschek-Wiener, C., Musen, M.A.: Mapping master: a flexible approach for mapping spreadsheets to OWL. In: The Semantic Web – ISWC 2010. LNCS, pp. 194–208 (2010). https://doi.org/10.1007/978-3-642-17749-1_13
28. Mulwad, V., Finin, T., Joshi, A.: A domain independent framework for extracting linked semantic data from tables. In: Search Computing, pp. 16–33 (2012). https://doi.org/10.1007/978-3-642-34213-4_2
29. Da Silva, A.R.: Model-driven engineering: a survey supported by the unified conceptual model. Comput. Lang. Syst. Struct. **43**, 139–155 (2015). https://doi.org/10.1016/j.cl.2015.06.001
30. Berman, A.F., Nikolaichuk, O.A., Yurin, A.Y., Kuznetsov, K.A.: Support of decision-making based on a production approach in the performance of an industrial safety review. Chem. Pet. Eng. **50**(1–2), 730–738 (2015). https://doi.org/10.1007/s10556-015-9970-x
31. Yurin, A.H., Dorodnykh, N.O., Nikolaychuk, O.A., Berman, A.F., Pavlov, A.I.: ISI models, mendeley data, v1 (2019). http://dx.doi.org/10.17632/f9h2t766tk.1
32. https://github.com/tabbydoc/tabbyxl/wiki/Industrial-Safety-Inspection
33. Dorodnykh, N.O., Yurin, A.Yu., Stolbov A.B.: Ontology driven development of rule-based expert systems. In: The 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC 2018), pp. 1–6 (2018). https://doi.org/10.1109/RPC.2018.8482174
34. Grishenko, M.A., Dorodnykh, N.O., Nikolaychuk, O.A., Yurin, A.Yu.: Designing rule-based expert systems with the aid of the model-driven development approach. Expert Syst. **35**(5), 1–23 (2018). https://doi.org/10.1111/exsy.12291