Chapter 2

Ontology for Application Development

Larysa Globa, Rina Novogrudska,
Alexander Koval and Vyacheslav Senchenko

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74042

Abstract

The chapter describes the process of ontology development for different subject domains for application designing. The analysis of existing approaches to ontology development for software platform realization in some subject domains is depicted. The example of ontology model development for telecom operator billing system based on descriptive logic is shown. For ontology model designing, it is proposed to use two formal theories: descriptive logic and set theory, which allow to systematize data and knowledge, to organize search and navigation, and to describe informational and computational recourses according to the meta-notion standards.

Keywords: ontology, descriptive logic, semantic information processing

1. Analysis of approaches to ontology designing

One of the existing approaches to the subject domain (SD) identification, based on the idea of conceptual modeling is ontological modeling. A conceptual domain model (CDM) describes the SD as a collection of concepts (terms) and relations between them. The entities from the real world correspond with the term of ontology and relations between such terms. This corresponds to the classical representation of the ontological model in which the ontology is defined by three finite subsets: concepts, connections, and interpretation functions. When a subject domain is modeling as a sphere of activity, the connections between concepts are also the terms that describe these relations. Concepts referred to a class of relations are used to describe the processes and phenomena of the real world. The conceptual model of the subject domain is defined as the totality of concepts (terms) and relations between them, which correspond to entities from

the real world, realized as an oriented labeled graph. The content model of the subject domain for the conceptual model is given by an oriented labeled graph whose vertices are interpreted as information elements corresponding to the real objects of the domain. Accordingly, two types of relations are defined in the models union: informative—to define the information element relation to another and conceptual—to define the relations of the element to the subject domain.

For a rapidly developing subject domain, the conceptual model is a constantly changing and developing structure. At the same time, the content model accumulates changes which over time lead to a modification of the conceptual model. The use of dynamic ontologies that are changeable in time will guarantee the actuality and adequacy of ontological models and, thereby, make them practically applicable to a wide range of tasks.

Ontologies are new intellectual tools for resources like Internet searching, new methods of knowledge, and queries presenting and processing. They can accurately and effectively describe the data semantics for a certain subject domain and solve the problem of concepts: incompatibility and inconsistency. Ontologies have their own processing facility (logical inference), corresponding to the tasks of semantic information processing. So, using ontologies, to execute searching request, the user will be able to receive in response resources that are semantically relevant to the query.

There are several approaches to the ontology concept definition, but there is no generally accepted definition. Depending on each specific task, it is convenient to interpret this term in different ways: from informal definitions to descriptions of ontologies in concepts and constructions of logic and mathematics.

Ontology is an attempt at a comprehensive and detailed formalization of a certain subject domain with the help of a conceptual scheme. Usually, such a scheme consists of a structure containing all relevant classes of objects, their relations, and rules (theorems, constraints) accepted in this field.

Ontology model advantages:

- Organization of semantic search
- Structuring of subject domain information.

For the systematization of information and computing resources, the ontological model can be used for such resources linking and description.

1.1. Advantages of using Semantic Web technologies

Nowadays, the search for advanced methods of information access, processing, presentation, and systematization is an important issue. The usage of ontologies reduces the time of computation and information retrieval, improves the efficiency of existing knowledge usage, performs logical deductions based on existing knowledge and integrates data from different sources using common semantics [1].

The sharing of information and semantics by people or program agents is one of the most common goals of developing ontologies.

Providing the possibility of SD knowledge reusing is an essential advantage of ontologies. To develop large ontology, it is in need to integrate several existing ontologies that describe parts of a complex subject domain. It is also possible to reuse the basic ontology and extend it to describe SD.

The analysis of knowledge in the subject domain is possible when there is a declarative specification of terms (that represent such knowledge). Formal analysis of terms is extremely valuable both when trying to reuse existing ontologies and when expanding them.

Thus, the usage of ontologies has several significant advantages. There is still a problem concerning searching the most complete method for ontologies development, with a view to their further usage. Based on large-scale projects, several approaches to ontology designing have been developed, but a single standardized method has not yet been selected.

1.2. Ontology development method

Several research groups offer methods for ontology development that were developed during the execution of their projects. However, these methods are different and none of them are standardized. Method of ontology development is an essential element in ontology designing to bring the process of development to the common, standard stages.

1.2.1. Approach Cyc

Cyc is the artificial intelligence project attempting to assemble ontology and knowledge base spanning the basic concepts and "rules of thumb" about how the world works [2]. The Cyc approach was formed during the execution of the project to develop a large base of general knowledge, which was executed in 1980s of the last century under the direction of D. Lenat. Within the framework of the Cyc project, the first tools of knowledge engineering were developed, and the knowledge representation language CycL, which was based, on the one hand, on the calculation of higher order predicates, and on the other hand, based at that time the language of the artificial intelligence systems Lisp. Within the framework of this project, the task of forming large knowledge bases, validation, and verification of such databases, as well as the task of knowledge-based deduction, was first posed and solved. In the framework of the Cyc-project, the idea of KB structuring in the form of microtheories was proposed, including knowledge from different areas, presented from different points of view.

Ontology development using Cyc project assumes the following phases [3]:

- "Manual" coding of explicit and implicit knowledge contained in knowledge sources.
- "Manual" knowledge encoding by means of programmatic facilities, using the knowledge that already exists in the Cyc.
- Semiautomatic phase, when the developer "recommends" the software tools for the source of knowledge for processing and "explains" to them the most complex places of processed texts.

At the same time, as a rule, two main tasks are solved at each phase:

- Development of a knowledge representation system and a top-level ontology that contains the most abstract concepts.
- Knowledge representation that was remained outside the formalization after the first task was solved on the basis of primitives, which was developed and implemented in the process of solving the first problem.

Currently, the Cyc KB already contains several hundred thousand terms and basic statements and millions of general knowledge statements derived from them. The Cyc KB fragment was recently released into open access called Open Cyc (one version of Open Cyc 2006 contains about 50,000 concepts and 300,000 facts), which is available to researchers in the field of artificial intelligence under the research Cyc license.

As Cyc approach applications, we can distinguish:

- System integration of heterogeneous databases, in which Cyc dictionary appears in the database schema, the resulting data from the database are interpreted in accordance with the terms of Cyc-ontology.
- An intelligent mechanism for searching images based on information contained in signatures to them.
- Module for the integration of structured terminology, which provides complex dictionary import, their integration and supports the corresponding management processes.
- The module for searching information on the Internet for the Cyc expansion.

1.2.2. Uschold and King's method

Uschold and King's method was proposed based on the results of the business process ontology modeling development that used enterprise ontology [4]. It was offered the methodology of designing that propose following stages:

1. Definition of purpose. Specification on why an ontology is developed, and how it will be used.

2. The ontology development. This stage is carried out in the following phases:

2.1. Ontology fixation, where occurs:

- detection (identification) of key concepts and relations,
- the development of precise textual definitions for each concept and relation,
- the identification of terms pertaining to each concept and relation,
- matching of all knowledge obtained in the process of developed ontology fixing.

2.2. The ontology coding. At this stage, the formal representation in the chosen language of knowledge representation is carried out.

2.3. At this phase, the possibilities of existing ontologies using and their integration into new ontology are realized.

3. The ontology evaluation. The stage is used for the developed ontology evaluation according to criteria such as:

   3.1. The ontology correspondence to the original goals and objectives

   3.2. Used software efficiency.

4. Documentation description.

The most important project using the method and methodology developed by Uschold and King was the Enterprise Project, which was carried out by the Artificial Intelligence Applications Institute of Edinburgh University with partners such as IBM, Lloyd's Register, Logic UK Limited, and Unilever, and the most important application of the method is the Enterprise Ontology development that is a collection of terms and definitions related to business enterprises.

With the use of Enterprise Ontology, the enterprise toolset toolkit was created that used the agent's architecture to integrate standard software products, serially produced in the plug-and-play style.

1.2.3. Gruninger and Fox methodology

Gruninger and Fox methodology was formed using the experience of specific ontology developing (using TOVE [5]) and focused on the subject domain of business processes modeling. This methodology provides the ontology development as a logical knowledge model and includes the following stages:

- Fixation of the motivational scenario. Within the framework of this methodology, it is postulated that the creation of any ontology is motivated by certain scenarios that arise in a particular subject domain, which specify a number of intuitively possible solutions for the problems indicated in the scenario.
- Formation of competence testing informal issues. The questions of ontology competence evaluation based on motivational scenarios are considered as requirements to the subject domain representation and the ability to solve problems specified in motivational scenarios with its help.
- Specification of the ontology terminology in the formal language, which is based on the following phases:
- Obtaining an informal ontology. As the result of ontology competence testing, a lot of terms are singled out that should be the basis for specification in the formal language.
- Specification of formal terminology. The terms identified in the previous phase are described in the formal language.
- Formulating questions of competence assessment using ontology terminology. At this stage, a specification of queries in the formal language to assess the competence of the ontology occurs.

- The specification of axioms for ontology terms in the formal language. Here, the semantics of ontology terms and restrictions on their interpretation are defined in the form of statements of first-order logic.
- Specifying the conditions for ontology completeness. At this stage, the conditions are set. Issues in solution implementation related to the competence of the ontology will be complete.

The most significant applied programs of Gruninger's and Fox's methodology are [3]:

- Enterprise Design Workbench is a designed environment that allows the user to analyze enterprise projects. An important functional feature of Enterprise Design Workbench is the support of an enterprises alternative projects comparative analysis.
- Integrated Supply Chain Management Project agent is the organization of the chain supply as a network of interacting intellectual agents. Every one of them performs one or more functions in the supply chain and coordinates its activities with other agents.

1.2.4. Methodology named "METHONTOLOGY"*

Methodology named "METHONTOLOGY" [6] was developed in the Madrid Polytechnic University laboratory. A distinctive feature of it is that METHONTOLOGY is formed on the basis of main activity analysis and rethinking for the activities inherent in the processes of software development and knowledge engineering. Thus, METHONTOLOGY integrates the experience of designing complex objects from two areas of knowledge.

This methodology includes the identification of the ontologies development process, the life cycle based on the prototypes evolution and individual techniques for performing each activity. The life cycle includes such stages as specification, conceptualization, formalization, implementation, and maintenance, as well as basic processes such as management, quality control, knowledge acquisition, integration, evaluation, documentation, and configuration management.

Examples of ontologies developed using METHONTOLOGY are [3]:

- CHEMICALS (contains knowledge in the field of chemical elements and crystalline structures).
- Monatomic Ions (collects information about monatomic ions).
- Environmental pollutant ontologies (represent methods to identify various polluting components in water, air, ground, and the maximum permissible concentrations of these substances, considering existing laws).
- The reference ontology (basic ontology for describing ontologies of "yellow pages" type directories).
- Silicate ontology (simulates the properties of minerals and silicates in particular).
- Ontologies developed in the IST-1999-2010,589 MKBEEM project (travel, textile catalogs, housing, used in the multilanguage e-commerce platform).
- OntoRoadMap (meta-ontology, ontology development methodologies, ontology development tools, ontology-related events (conferences, seminars, etc.)).

Examples of applications that use some of the above ontologies:

- (Onto) Agent (an ontology broker that uses reference ontology as a source of knowledge and finds a description of the ontologies that satisfy the given constraint).
- OntoRoadMap application developed as (Onto) Agent. This is an ontology-based web application that allows the community to register, view, and find ontologies, methodologies, software tools and languages for ontologies development, programs in Semantic Web, e-commerce, NLP, etc., as well as major conferences, seminars, and events in these fields).
- Ontogeneration (a system using the ontology CHEMICALS and the linguistic ontology of GUM to generate texts in Spanish in response to a query in the field of chemistry).

1.3. Comparative characteristics of methods for ontology development

The methods of ontology development described above were comparable in the following parameters:

- Terms of use. Shows the necessary conditions of use for ontology designing proposed in the ontology development method.
- Development process. It shows the method's specific features for ontology development and its stages.
- The implementation process.
- Preservation and use. This stage shows whether the proposed method further preserves and uses the developed ontology.
- Knowledge obtaining. This stage shows whether the method described the possible knowledge ontology acquisition.
- Ontology control and confirmation.
- Ontology configuration management. This stage shows whether the method described controlling the ontology configuration.
- Ontology documentation.

The comparison results of considered ontology development methods by these parameters are given in **Table 1**. The sign (+) means "described in detail," (+/−) means "interrupted," and (−) means "not parsed."

1.4. Methods of semantic web usage for data warehouses development

Data storage (DS) provides multidimensional view of a huge amount of historical data from operational sources; thus, they provide useful information that allows decision makers to improve business processes in an organization. Multidimensional models allow you to structure information into facts and measurements. The fact contains the necessary dimensions (attributes of the fact) of the business process (sales, deliveries, etc.), while the dimension is a context for the analysis of facts (product, client, time, etc.)

	Cyc	Uschold and King's	Gruninger and Fox	METHONTOLOGY
Terms of use	—	—	+	+
Development process	—	—	+/−	+
The implementation process	+/−	+/−	—	+
Preservation and usage	—	—	—	+/−
Knowledge obtaining	+/−	+/−	+/−	+
Control and confirmation	—	+/−	+/−	+/−
Ontology configuration management	—	—	—	+/−
Documentation	+/−	+/−	+/−	+

Table 1. Comparative characteristics of ontology development methods.

Data storage contains information that is specified for data analysis. This information is obtained from existing OLTP databases and is preprocessed to synchronize syntax and semantics. Thus, one of the main goals of data warehouses is the integration of information obtained from different sources. After that, OLAP systems can be used for efficient use of stored information. Both types of systems use multidimensional data models [7].

Integration of information coming from different sources is one of the main goals of data storages. The Semantic Web technology can be used to develop data storages since data structures depend on the context to determine the actual data semantics and to provide context-sensitive knowledge [8].

Semantic Web is a source of knowledge, the exploitation of which will open new opportunities for academic and business tasks. One such feature is the analysis of information resources to support decision-making, such as the trends identification and the discovery of new influence factors. Semantic annotations are a formal information resource description that is usually based on common ontologies of SD [9]. The main reason for using domain ontologies is to develop common terminology and logic concepts that are available in a specific SD.

The topic of ontologies used for the data storage development was partially covered by other authors in some aspects such as ETL processes [10] or data sources [11].

Data storage designing and development uses ontologies in the following cases.

1.4.1. Requirements analysis

In order to reach the understanding of designing requirements by all project participants, ontology implementation is very important. Participants may have incompatible needs; thus, it is especially useful [12]. It is convenient to manage business models in the form of ontologies because they represent a descriptive abstraction of the environment in which the software (including data storage) should work. They also contain semantics that have already been agreed upon with the stakeholders in the process. In addition, ontology can be used as an auxiliary tool for information requirement analyses [13].

1.4.2. Needs and data source matching

One approach to reconciling the needs and sources of data is closely related to the idear of ontology using to obtain multidimensional data [11]. It is a method for detecting a multi-dimensional structure from ontologies. This method consists of:

- identifying facts and measurements by matching and categorizing,
- selecting the view and determining the measurements for the facts,
- determining the basis for the search and filtering,
- and determining the aggregation hierarchy by identifying the relations between entities. Elements of measurement are determined using heuristic procedures based on structural aspects (such as cardinality and selectivity). Another approach to use heuristics is given in [14]. The scientific community should work on providing both semantic rules and (public) repository of multidimensional annotated ontologies. At the end, the ontology of Cyc, or its open source version of OpenCyc, is interesting.

1.4.3. Data types in dimensions

Defining data types at the ontology level in systems with multidimensional models allows data storage designers to correctly describe model sample with the necessary data types in dimensions. Many existing subject domain ontologies can help developers to design OLAP systems in accordance with generally accepted requirements.

1.4.4. Incomplete input data

The data sources used in the SD may contain not all the necessary information. An additional set of data can be obtained from other publicly available sources of information. For example, commonly used ontologies, such as WordNet (lexical base), can be used to fill elements that are not supported in data sources. Another example is the Computing Classification System Taxonomy, which can be used for computer classification and obtaining more detailed aggregation of measurements in data repositories [15].

1.4.5. Logical output when querying OLAP systems

Requests to OLAP systems are based on manipulations with aggregation data. However, the algebra of OLAP systems is based on computations instead of logic. Any statement to the account of a multidimensional model cannot be proved, but only calculated.

The main advantage using ontologies in data repositories is the extension of OLAP requests with the possibility of logical output.

Requests for OLAP systems are based on manipulations with aggregation data. However, the OLAP systems algebra is based on computations instead of logic. Any statement to the account of a multidimensional model cannot be proved, only calculated.

The main advantage of ontologies using in data storages or data repositories is the extension of OLAP requests with the possibility of logical output.

2. Ontology model of the billing system

2.1. Descriptive logic for ontology development

One of the main tasks of the ontology development is the definition of the descriptive logic that will be used. It is necessary to determine the solvability and computational complexity of the logic used, characterizing the possibility and speed of obtaining logical inferences from an ontology. Based on the chosen logic, the ontology definition language (OWL-DL, OWL-Lite, OWL-FULL) and the corresponding tools for implementation are selected.

Descriptive logic (DL) is a family of logics with different capabilities. Accordingly, for different tasks depending on the purposes, different DL logics are chosen. For example, the OWL Lite language is based on a DL called SHIF (D), and OWL-DL is based on SHOIN (D) (the explanations of the abbreviations will be explained below).

Any DL is a subset of FOL (first-order logic). This means that any statement on DL can be represented as an FOL formula (but not vice versa). At the same time, they are semantically compatible, that is, if you turn the knowledge base of DL into a knowledge base of FOL, then it will be possible to draw the same logical conclusions from it as to the transformation.

DL syntax does not explicitly use variables and quantifiers. For example, the statement $A \subset B$ in DL is the same as the FOL-formula $\forall x\ A\ (x) \rightarrow B\ (x)$, but without variables.

Most DL logic is usually solvable that is achieved by cutting some FOL capabilities (in particular, variables). It should be noted that OWL DL is solvable and there are logical processors for it, and OWL Full, which is not based on DL, does not exist and there are no logical processors for it [15].

DL logic combines rich expressive possibilities and good computational properties such as solvability and relatively low computational complexity of the main logical problems that make their application possible in practice.

ALC logic is one of the main descriptive logics, which is basic to many others. For many real ontologies, ALC logic is enough.

The ALC language contains the alphabet (that is, the set of base characters) that consists of three components:

- A set of base-class names (NC) and two special classes (top or universal class and bottom-empty class)
- A set of relations names (NR)
- A set of instance names (NI).

The central feature of ALC, as many DLs, is the ability to describe complex classes (concepts). This is done with the help of the following statements called class constructors [16]:

- Class crossing or conjunction ($C \cap D$)
- The union of classes or the disjunction ($C \cap D$)
- Addition of class or negation ($\neg C$)
- Universal constraint ratio ($\forall RC$)
- Existential restriction ratio ($\exists RC$)
- Logical formulas in ALC are called axioms. There are three types of axioms:
- Relations of the "class-subclass" type. These axioms have the form "$C \subset D$," where C and D are arbitrary (possibly complex) classes.
- Relations of the type "class individual" They have the form "*a: C*," where "*a*" denotes an object and C is an arbitrary class.
- Relationships of the type "relationship individual" type. They have the form "(a, b): P," where "a, b" denote two objects and P is an arbitrary relation.

A set of axioms of type 1 is called TBox (abbreviation of terminological box). A set of axioms of types 2 and 3 is called *ABox* (assertional box).

The knowledge base (or ontology) in the ALC is a collection of *TBox* and *ABox. TBox* is actually a description of the class hierarchy (domain concepts). *ABox* is a collection of facts about specific objects, to which classes they relate to and what relations they have.

At the heart of the ALC semantics, there are two key components: domain Dom and the interpretive function *I*. Dom is a limited set of elements (sometimes called "real world" elements) and is defined as follows:

- Each base class in NC is linked to a subset of *Dom*, with I (*top*) = *Dom* and I (*bottom*) = *empty set*
- Each relation in NR is linked by some relation to *Dom*.

(the subset of *Dom x Dom*)

- I maps each object in the NI per element in *Dom*.
- In other words, $I(C) = X$ means the following: "The symbol C denotes the set of elements X of the real world."
- Interpretation of complex classes [16]:
- Interpretation $C \cup D$ (or $I(C \cap D)$) is equivalent to $I(C) \cap I(D)$
- $I(C \cup D) = I(C) \cup I(D)$
- $I(\neg C) = Dom \neg I(C)$

- $I\ (\forall\ R.C)$ = all such $x \in Dom$ that for any $y \in Dom$ it is true that

$(x, y) \in I\ (R) \rightarrow y \in I\ (C)$

- $I\ (\exists\ RC)$ = all such $x \in Dom$, that there exists $y \in I\ (C)$ such that $(x, y) \in I\ (R)$

Explanation: For example, the precedence formula defines the following simple meaning for the constructor: *RC*: "If class *X* is defined as ∀ *RC*, then *X* denotes the set of all objects *x* such that all objects associated with them for the relation R are elements of class *C*."

- Axioms of logic:
- Interpretation *I* satisfies the axiom $C \supset D$ if $I\ (C) \subset I\ (D)$.
- *I* satisfies the axiom *A*: $C \in D$ if $I\ (a) \in I\ (C)$.
- *I* satisfies the axiom (x, y): *P* if $(I\ (x), I\ (y)) \in I\ (P)$.

If interpretation *I* satisfies some axiom *A*, then it is called the model *A*. Interpretation *I* satisfies the *TBox* (or is a *TBox* model) if it is a model of all the axioms of the *TBox*. The *ABox* model is similarly defined. An interpretation is a model of ontology if it is a model of its *TBox* and *ABox*.

Using the domain and the interpretive function, "formally" defines the meaning of classes, objects, relations, and logical formulas (axioms).

Let us describe logical output in ALC. Knowledge bases are formulated in the language of descriptive logics, they are used not only to represent knowledge of the SD, but also for the logical analysis of knowledge, that is to check the absence of contradictions in them, the withdrawal of new knowledge from existing ones, and the ability to make inquiries to knowledge bases. Due to the fact that knowledge bases of DL are written in formalized form, it is possible to make a strict logical conclusion. As the syntax and semantics of the descriptive logics are developed in such a way that the basic logical problems are solvable, the derivation of new knowledge can be developed by computer means—output machines.

For an ALC, you can see the basic tasks of the logbook [16]:

- Consistency (or noncontradiction) ontology. An ontology is coherent if it has at least one model. In other words, if there is such a way of interpreting (i.e., assigning meaning) classes, objects, and relations that do not conflict with any of the given axioms (TBox or ABox).
- Class feasibility. *A* class *C* is called coherent in the ontology of *O* if at least one model O interprets it as a nonempty set.
- The conclusion of the axioms. From the ontology *O*, an axiom is derived, and each model of *O* is also a model of *A*. In other words, if the interpretation does not contradict the axioms in *O*, it does not contradict the *A*.

Important is the fact that the two last tasks are reduced to the task of coherence in the following way:

- Class *C* is coherent in the ontology of *O* if and only if the addition of a new axiom *A*: *C* (where the object "*a*" has not previously met in *O*) does not lead to a loss of consistency. That is, the issue of coherence *C* is solved by adding a new axiom to *O* and checking consistency.

- Axiom *A* is derived from the ontology of *O* if and only if adding the "negation" of axiom *A* to the ontology of *O* leads to a loss of consistency.

Here, it is necessary to determine what denial of the axioms is. For the axiom "$C \supset D$," the axiom *A: (C* ∪ ¬ *D)* will be denied, and for *A: C,* the denial will be a: ¬ *C*.

This is a classic method of proof "from the opposite." If the addition of the negation of the statement leads to contradiction, then the statement is true (that is, it logically follows from the ontology).

Practical importances are nonstandard algorithmic problems, in particular [17]:

- Classification of terminology: for this terminology (i.e., TBox) to develop a taxonomy or hierarchy of concepts needs to arrange all atomic concepts concerning meaning (relative to a given TBox).
- Extraction of concept copies: find all instances of a given concept based on a given knowledge base.
- The narrowest concept for an individual (instances): find the smallest (by attachment) concept, an example of which is a given individual with respect to a given knowledge base.
- Response to knowledge base query: give out all sets of individuals that satisfy a given query to a given knowledge base. Conjunctive queries to knowledge bases (and also their disjunctions), which are similar to queries from the field of databases, have been extensively studied.

2.1.1. Extension of ALC logic

There are numerous extensions of ALC logic to additional constructors for concepts, roles, and additional axioms in *TBox* description.

The most famous extensions are [18]:

F: Functional roles: concepts of the form (≤ *1 R*) that means: there is no more than one *R* follower

- *N*: limitations of cardinal roles: concepts of the view (*R* ≤ *n*) that means: there are no more *R* followers.
- *Q*: qualitative limits of cardinal roles: concepts of the form (≤ *n R.C*) that means: there are no more *R* followers in *C.*
- *I*: reversal roles: if *R* is a role then R^{-} also has a role that means the treatment of a binary relation *O*.

Nominees: if *b* is the name of an individual, then *{b}* is a concept that means a single-element set.

- *H*: hierarchy of roles: in the *TBox*, the axioms of the nesting of the roles $R \subseteq S$ are allowed.
- *S*: transitional roles: in the *TBox*, the axioms of transitivity of the form *Tr (R).*
- *R*: The axioms components of the role nesting in the *TBox* ($R \bigcirc S \subseteq R$, $R \bigcirc S \subseteq S$) with the condition of acyclicity where $R \bigcirc S$ represents the composition operation for roles Language Extensions by specific domains (data types).

2.2. Stages of ontology development

On the basis of the methods discussed in the first section for ontology development, the following steps were selected:

- Defining goals, scope of application of ontology, and set of terms. Ontology of billing is developed as an integral part of the general ontology of OSS/BSS systems for the integration of information from various sources.
- Define classes and develop a hierarchy.
- Definition of relations.
- Limitations and relations of properties. In determining the constraints and properties, available ontology ratio is also determined by the descriptive logic that is required for this ontology.
- Creating instances of classes.

Below is a detailed description of each stage on an example of the ontology development of one of OSS/BSS components—the billing system.

2.2.1. Definition of goals, scope, and set of terms

To determine the goals and scope of application of ontology, it is necessary to answer the following questions [19]:

- What is the SD that the ontology reflects? What will be used? What types of questions, the information presented in the ontology, must be answered and who will use this ontology?
- Billing is an automated system of accounting for the services rendered, their billing and invoicing for payment. Billing system is an important element of software for any operator activity. Ontology of billing system was developed as an integral part of the general ontology for OSS/BSS systems.

At an early stage, it is important to develop a complete list of terms, including concepts that overlap and duplicate.

2.2.2. Define classes and develop a hierarchy

Classes in the developed ontology should be close to physical or logical objects, and their relations to the relations of these objects.

There are several approaches to creating a hierarchy of classes [20]:

- From up to down. It starts with defining the most general concepts of the domain and further detailing the objects in the hierarchy.
- From bottom to the top. It starts with defining detailed and specific classes (the end of the hierarchy tree) with further grouping into more general concepts.

- Combination of the first two methods. It consists in the creation of objects that are completely understandable and then group them into groups and develop more specific objects.

There are six main classes: "Provider," "Service," "Tariff plan," "Account," "User," and "Account." There are also three subclasses for the "Service" category: "Telephony," "Internet," and "Television" and three subclasses for the "Account" class: "Receipt," "Balance," and "Bill of Invoice."

- Class "Provider" contains information about the service provider.
- Class "Service" contains information about the services provided by the provider and their description. It is a parent for subclasses of Internet, Television, and Telephony services.
- Class "Tariff plan" contains information about the tariff plans of the provider.
- The "Account" class is a user account (person).
- The "User" class is direct information about a person who has an account (his/her name, address, etc.).

The "Account" class contains information about the cash accounts associated with this user account. This class is a parent for the "Received" (Receipt), Balance, and "Invoice" classes.

2.2.3. Relations definition

There are two types of relations: the relations between classes and the relations between data types. You can also classify the relations by the following types:

- Internal relations: these are the relations that are inextricably linked with the object.
- External relations are those relations that describe the connection of objects with external objects.
- Relations between instances of this class.
- The relations between instances of different classes from different parts of the hierarchy.

The class "Provider" is associated with the "Service" class with the "has a Service" relation, with the Tariff Plan class relation "has a Tariff Plan", and with the "User" class the relation "has a relation." The class has data-type relations:

- "has a name" with the data type string (the name of the "Provider");
- "has a date of creation" with a data type dateTime (the date of "Provider creation");
- "Has a description" with the data type string (additional description of "Provider").

The "Service" class has the following data type relations:

- "named" with string data type (name "Services")
- "has a description" with the data type string (description of "Services").

The "Tariff Plan" class is related to the "Service" class with the "has Service" relation. It has the following data type relations:

- "named" with string data type (name "Tariff Plan");
- "has a description" with the data type string (description of "Tariff Plan");
- "has value" with the data type integer (the value of "Tariff Plan").

The "Account" class is related to the "Account" and "Tariff Plan" classes in relation to "Has a Bill" and "Uses the Tariff Plan," respectively. It also contains the following data type relations:

- "has login" with data type string (Login "Account");
- "has a password" with the data type string (Account password);
- "has an email" with the data type string (registered e-mail "Account").

The class "User" is associated with the "Account" class with the relation "has an Account," with the class "Service" with respect to "using the Service" and with the "Provider" class, the relation "has a relation." In the class description, there are following relations of data types:

- "has a full name" with string data type (username "User");
- "has passport data" with string data type (Passport data of "User");
- "has address" with data type string (address of "User").

The "Account" class is associated with the "Tariff Plan" class for the "has a Tariff Plan."

The "Payed Invoice (Receipt)" class has the following data type relations:

- "has a paid amount" with the data type integer (paid amount)
- "has a payment date" with the dateTime (date of payment for this account).

The "Balance" class has the following data type relations:

- "has a sum" with the type of data integer (amount of money on the balance sheet)
- "has a balance date" with the dateTime (date for which the balance information is viewed).

The main classes of ontology and the basic relations between them are depicted in **Figure 1**.

2.2.4. Relation limitations and relation properties

Restrictions on relations allow you to describe the valid values, type, number of values, and other features that a property of this class can possess. Relations can be symmetrical, reflexive, traceable, reversible, may have functional limitations, or be hierarchical.

A detailed description of the constraints and relation properties available on this ontology is given in section "Definition of descriptive logic."

2.2.5. Creating class instances

The validation of the developed ontology and the creation of specific instances of classes are performed at this stage.

The important question is: "Whether to develop a new subclass or make a new value an instance of an existing class?"

There are several rules that can be used to answer this question [21]:

The subclass of an existing class is usually:

- Has a relations that a superclass does not have.
- Has other restrictions on the ratio, unlike the restrictions in the superclass.
- Participates in other interactions between classes, unlike the superclass.

2.3. Definition of descriptive logic

To determine descriptive logic and, the writing language of ontology, which is necessary to describe this ontology, it is necessary to determine which types of relations and constraints are present in this ontology:

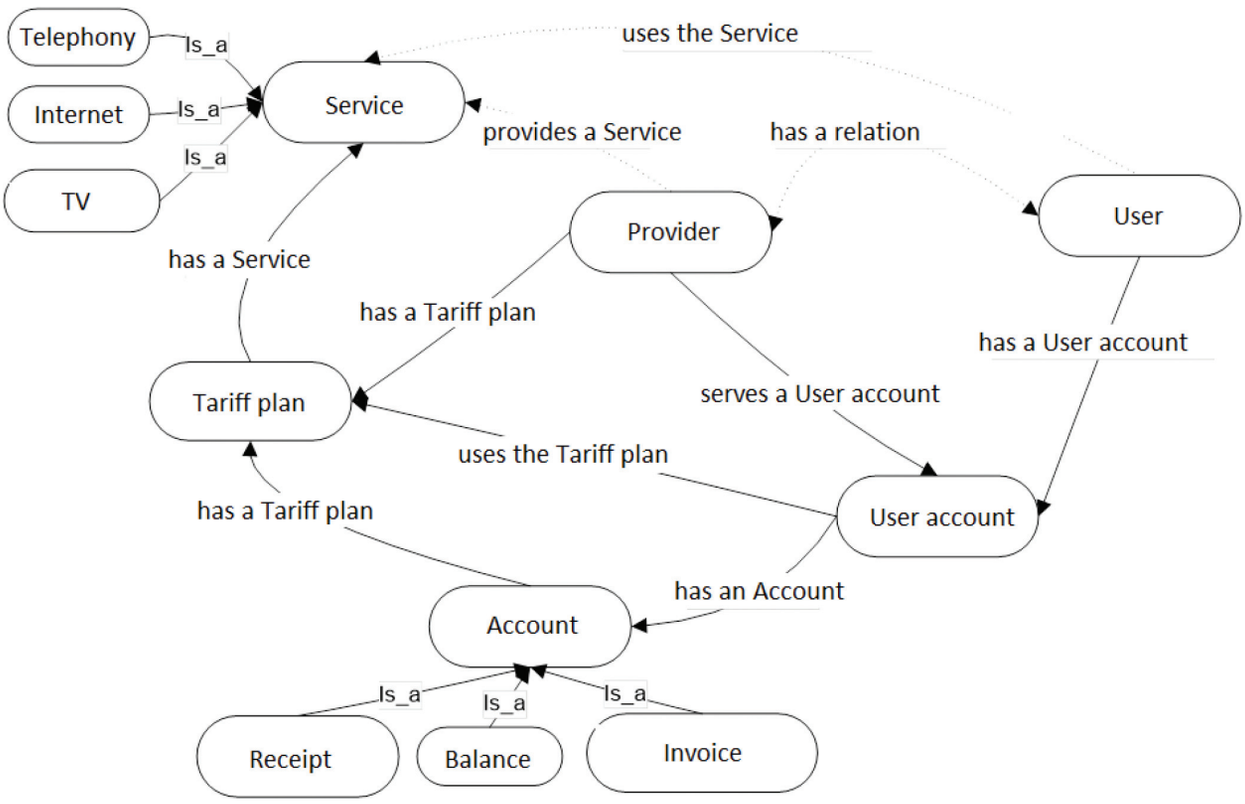

Figure 1. Billing system ontology.

- Inversion of the relations $R \equiv S-$. Many relations in ontology have explicitly defined reciprocal relations. This relation has a "Bill" and "is a Bill," "has an Account" and "is an Account," "has Service" and "is a Service," and "has a Tariff Plan" and "is a Tariff Plan." Reverse relations are convenient to avoid errors when filling in information. If one of them is present in ontology, inverse relations will be listed automatically with the output machine. It is also possible to use implicitly given inverse relations. For example, to describe the relations "has a relation," an inverse "serving account" was used.
- Relations of hierarchy $R \subseteq S$. In general, these relationships correspond to the hierarchy of classes. The relationship "has an Internet service," "has a Service Telephony," and "has a TV service" is a subsidiary of the "has Service" relationship. The ratio is "Balance," "Has a Bill of Account," and "Has a Paid Account", a subsidiary of the "Has Account." As a result of the deduction, if there is a child relationship in the ontology, the output machine will be added to the relation for the parental relation.
- The symmetry of relations $R \subseteq (R^{-})$ and the asymmetry of the relations $R \subseteq$ not (R^{-}).

The relation "has relations" between the classes "User" and "Provider" is symmetric. Most of relations are asymmetric, for example, the ratio "has a Tariff Plan," "serves the Account," etc. For symmetric relations, the relation for the inverse pair is automatically added during derivation. For preventing errors when filling in information in an ontology asymmetry for relations is indicated.

- Functionality of relations ($\leq 1\ R$). Most typified relations are functional, for example, the "has a password" relations for the "Account" class or "has a creation date" for the "Provider" class.
- Irreflexivity of relations. In the absence of reflexive relations, all relations of this ontology are irreflexive. This attribute property is added to prevent errors that may occur when filling in information in an ontology.
- Composition of relations R o S. It is used to determine relations "has a relation" defined as the composition of relations "has an Account" and "inverse servicing Account," and to determine the relation "uses the Service," defined as the composition of relations "has an Account," "Uses the Tariff Plan," and "has a Service." These ratios are added by the output machine if the ontology has the necessary chain of relations.
- Data type relations

For this ontology, the expression of the SRIF (D) is very clear, where the letters SR mean transitivity, composition, and characteristics (symmetry, reflexivity, etc.) roles, I—inversion, F—functionality, and (D)—the ratio of data types. Since the ontology uses the composition of the relations and they were added only in the version of the language describing the ontology OWL 2, this language was chosen for the further ontology development.

2.4. Managing metadata in data storage

Data vault implementation is a complex task that requires developers and architects to have sufficient knowledge in the SD and appropriate expertise. The use of ontology can be useful in many aspects of designing data storage.

Semantic Web allows companies and organizations to store and process a huge amount of valuable semantically annotated data. To date, many applications attach metadata and semantic annotations to the generated information (using domain and application ontologies ontology).

Consider, for example, a customer relationship management (CRM) system in which customer addresses are stored, and a billing system in which we can find information on account receipts, debts, etc. Transferring this information to the data vault, the ontology can set up the necessary links to provide combined customer information, as well as for further analysis (for example, the most risky and unreliable locations).

In order to be able to manage complex processes and large volumes of data, data warehouse system (DWH) should be supplemented with additional information. The framework for managing metadata, using the metadata repository as the basis, allows you to effectively develop and store such metadata. **Figure 2** depicts the general architecture of the data store. Data come from heterogeneous internal and external data sources and integrate into data storage for further analysis.

Since different data sources have heterogeneous data models that are different from the data storage model, data transformation needs to be implemented to ensure structural and semantic compliance. The ETL process between the levels of data sources and data management ensures appropriate extraction, clearing, transformation, and downloading of data from sources to the SD. The upper level of architecture is analysis level that provides various analytical tools [22].

All levels of architecture are connected to the metadata repository, which stores metadata for all four levels. The metadata store contains information that is needed to support the administration, development, and use of the data storage.

Data storage metadata can be used for different purposes and include different types of information. The two main areas of application are tracking the origin of the data and analyzing the data interconnections. Tracking the origin of the data allows the business user to track the data elements within data storage or other systems that are data sources. The data impact analysis is used to identify the potential impact of planned changes on some data elements in the DWH or the source system before they are actually implemented. Requests for this type are usually processed in graphs that show how data elements are converted to different levels of data storage. In order to analyze these graphs, using relational database tools and SQL queries, complex algorithms are required. Regarding this, some commercially available metadata management tools (such as ASG Rochade) do not open access to their mechanisms.

Semantic Web technologies, the Resource Description Framework, OWL (Web Ontology Language), OWL (Web Ontology Language), program logic modules, rules for adding rules, such as SWRL (Semantic Web Rule Language), as well as SPARQL query language, provide proven, standardized management tools structures of graphs. In addition, mapping using ontologies can be used to manage, store, and integrate metadata in heterogeneous data storage systems.

Data storage is used to support users in achieving more efficient and fast business decisions or to open business trends. Data are extracted from various sources of internal and external data

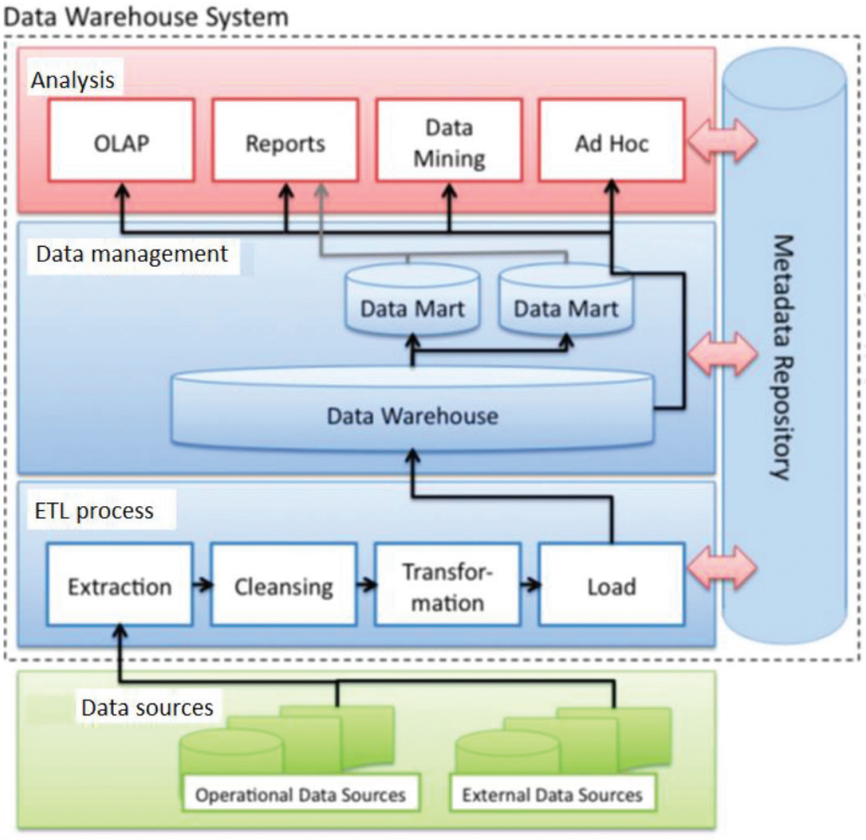

Figure 2. The data store general architecture of the billing system.

and historically integrated into the data storage, serving as the basis for analytic queries. Since different data sources have heterogeneous patterns and structures that are different from the data storage data model, transformation is required in order to eliminate structural and semantic differences.

In **Figure 3**, the ETL process is depicted as the level between data sources and the level of data management. It covers copying, clearing, converting, and downloading data to data storage. The level of data analysis that reflects various analytical systems, such as OLAP and data mining, is placed above the data management level. All data storage levels are connected to a metadata repository, which stores the metadata of each component. The metadata contains information that supports the operation, development, and administration of data storage.

Comprehensive, manageable metadata storage improves the extraction of information, reduces efforts to develop and administer [23]. The metadata include the logical and physical models of the data of individual components and their relations. With this information, two of the main metadata management tasks—the origin of data and impact analysis—can be addressed. Data origin requests are used to trace the path of certain data elements through various DWH components, for example, a user who has requested a certain metric and wants to know which system the attribute starts with [24]. Impact analysis is aimed at identifying the possible effects of modifications of data elements, showing which other data elements will also be affected by this change.

http://dx.doi.org/10.5772/intechopen.74042

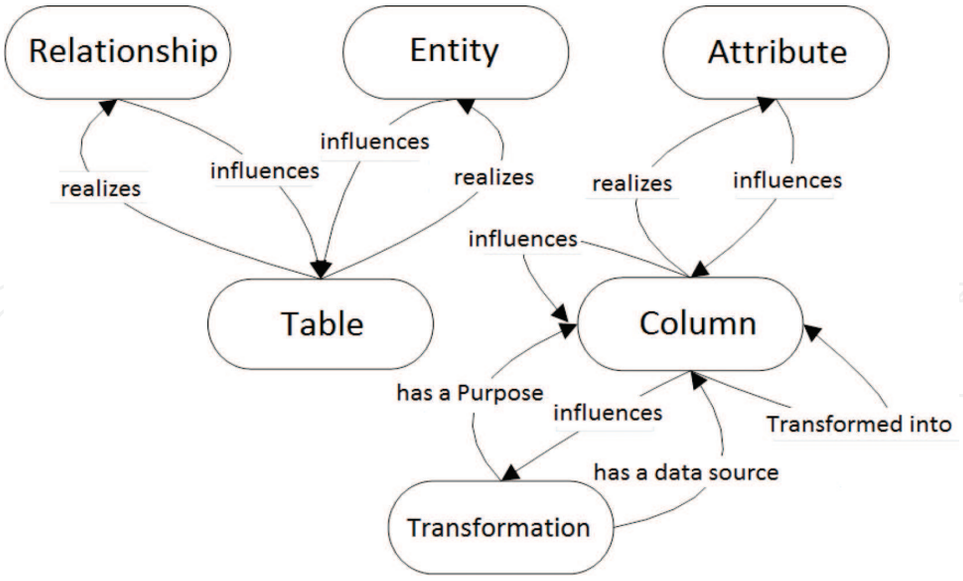

Figure 3. The data storage metamodel.

Most ETL software has a metadata management system (e.g., Informatica PowerCenter, IBM DataStage, Microsoft SQL Server Integration Services). Also, these vendors provide tools for importing and exporting metadata.

There are also decisions that relate directly to subject domain and do not relate to individual ETL programs. In this case, you need to look for a way to download them to the system. However, the mechanisms used in these cases are commercial secrets (for example, in ASG Rochade use the internal procedural language Rochade Procedural Language (RPL)) or based on a relational database.

Since elements of different levels of data vault are connected through transformation, these connections can be described as a graph. The main task of the metadata management system in this case is an effective analysis of queries in these graphs. For these purposes, it is proposed to use Semantic Web technologies: RDF, OWL, software for logical output to improve the efficiency of queries.

The DWH metadata model should be displayed as an OWL ontology, in which the metadata itself will be stored as an ontology instance. Using OWL language features and using language to create rules for SWRL, logical output can be obtained by a more complete ontology. Requests for these data can be done using the SPARQL language.

Let us describe the example of a metadata model (**Figure 3**). In the model, the following logical elements are present: *Relations, Entity, Attribut*e, physical elements: *"Table," "Column,"* as well as a separate *"Transformation"* element that reflects the transformation of some columns of tables into others. There is a detailed description of the items below:

Class "*Relations*" is used to store information about the relations between entities in the data model. As the connections example, it is possible to use the external key. This class can be used with the attitudes "*influences*" and "*realizes*" with the class "*Table.*"

Class "*Entity*" is used to represent objects of the subject domain. The "*Table*" class implements physically this class. This class may have relations "*realizes*" and the relations "*influences*" with the class "*Table.*"

Class "*Attribute*" logically reflects the objects attributes of the subject domain. The class "*Column* implements physically this class, but it can relate to relations "*realizes*" and "*influences*" with this class.

Class "*Column*" is a physical implementation of the "*Attribute*" class. It is a base for the "*Transformation*" class and may have such relations as "*influences,*" "*realizes,*" and "*transforms.*"

Class "*Table*" is a physical realization of the classes "*Entity*" and "*Relations*" and is a domain class for the ratio "*realizes*" for these two classes.

Class "*Transformation*" is additional for the metadata model and is designed to display the relations between the columns of data sources and data storage columns. It may have relations "*has_a goal*" and "*has a data source.*"

The logical element implementation in physical level is displayed using the ratio "*realizes.*" This model is implemented as OWL ontology. Also, in the model, there are relations between the columns "*transforms_into,*" which is defined by the "*has_a_goal*" and "*has_a data_source*" data.

The ratio of "*influences*" is defined by the ratio "*realizes*":

- has a data source (? *t.*? *a*) ∧ has a goal (? *t.*? *b*) → transforms (? *a*?, *b*)
- realizes (? *a*?, *b*) → influences (? *a.*? *b*)
- realizes (? *b.*? *a*) → influences (? *A.*? *B*)
- has a data source (? *T.*? *A*) → influences (? *A.*? *T*)
- has a goal (? *T.*? *A*) → influences (? *T.*? *A*).

The used Turtle code to describe the properties of relations data is given in **Figure 4**.

```
#################################################################
#    Object Properties
#################################################################
:influence rdf:type owl:ObjectProperty ;
      owl:propertyChainAxiom ( [ owl:inverseOf :has a data source] ) , ( :realizes) ,
(:has a goal) , ( [ owl:inverseOf :realizes]) .
:has a data source rdf:type owl:ObjectProperty ;
:has a goal rdf:type owl:ObjectProperty ;
:realizes rdf:type owl:ObjectProperty .
:transforms_в rdf:type owl:ObjectProperty ,
                    owl:TransitiveProperty ;
           owl:propertyChainAxiom ( [ owl:inverseOf :has a data source]:has a goal)
```

Figure 4. Example of the Turtle code.

It becomes possible to obtain new data due to logical deduction using the proposed model. By specifying, for the data source columns, additional metadata that will indicate in which transformations these speakers are involved, it will be possible to trace the data path from the sources to the data storage. Also, the use of this model makes it possible to detect the impact on the system of changing certain components of the data vault.

3. Summary

1. An analysis of the feasibility of ontology development is conducted, and the main advantages of using Semantic Web technologies in information systems are given.
2. A detailed review of existing methods of ontology development is carried out, and features, priorities, and drawbacks of each of them are given, and a comparative description of methods of ontology development is given.
3. Examples of applications of Semantic Web technologies at different stages of development and use of data warehouses, namely, for requirements analysis, reconciliation of needs and data chains, determination of types of data in measurements, and incompleteness of input data are considered.
4. The semantics and syntax of the main descriptive logic ALC was described, and the basic axioms, extensions, problems of logical conclusion, and nonstandard algorithmic problems were defined.
5. The stages of the ontology of the billing system were defined and described, the basic classes were defined, the hierarchy and relations between classes were described, and class instances were created for checking ontology for incompatibility and obtaining output.
6. The method of metadata management with the use of Semantic Web technologies is proposed, the metadata ontology is described, the logical and physical classes are described, and the relations between them are determined.
7. For the proposed metadata ontology, the properties of the relationships between classes were described, which in the future would allow new data to be extracted as a result of logical output, as well as in listing the Turtle code to describe these relationships.

Author details

Larysa Globa[1]*, Rina Novogrudska[1], Alexander Koval[1] and Vyacheslav Senchenko[2]

*Address all correspondence to: lgloba@its.kpi.ua

1 National Technical University of Ukraine, Kyiv Polytechnic Institute, Kyiv, Ukraine

2 Institute for Information Recording, Kyiv, Ukraine

References

[1] Голиков НВ. Применение онтологий/Н.В. Голиков//Институт вычислительных технологий СО РАН, Новосибирск [Електроний ресурс] Електрон. текстові дані. Режим доступу: http://www.ict.nsc.ru/ws/YM2006/10628/golikov.html Середа. 15 квітня 2014

[2] Cyc, access address: https://ru.wikipedia.org/wiki/Cyc, access data: 5.12.2017

[3] Ефименко ИВ. Онтологическое моделирование: подходы, модели, методы, средства, решения. И.В. Ефименко, В.Ф. Хорошевский [Електроний ресурс] Електрон. текстові дані. Режим доступу: http://www.hse.ru/data/2011/12/22/1261631357/WP7_2011_08_1.pdf Понеділок. 17 березня 2014

[4] Dietz J. Enterprise Engineering Enterprise Ontology [Електроний ресурс] Електрон. текстові дані. Режим доступу: http://www.siks.nl/map_IO_Archi_2006/J.Dietz.pdf Вівторок. 6 травня 2014

[5] TOVE Ontology Project [Електроний ресурс] Електрон. текстові дані. Режим доступу: http://www.eil.utoronto.ca/enterprise-modelling/tove/Неділя травня; 2014

[6] Ferndndez M. METHONTOLOGY: From Ontological Art Towards Ontological Engineering./ M. Ferndndez, A. Gomez-Perez, N. Juristo [Електроний ресурс]. Електрон. текстові дані. [Режим доступу: http://oa.upm.es/5484/1/METHONTOLOGY_.pdf Субота. 3 травня 2014

[7] Rizzi S, Abelló A, Lechtenbörger J, Trujillo J. Research in data warehouse modeling and 20 design: Dead or alive. In: Song I-Y, Vassiliadis P, editors. DOLAP '06: Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP. New York, USA: 2006. pp. 3-10

[8] Santini S. Ontology: Use and abuse. In: N. Boujemaa, M. Detyniecki, A. Nu rnberger, editors. Adaptive Multimedia Retrieval, Vol. 4918 of Lecture Notes in Computer Science. Spring; 2007. pp. 17-31

[9] Oren E, Moller K. What are Semantic Annotations/Digital Enterprise Research Institute, National University of Ireland, Galway [Електроний ресурс] Електрон. текстові дані. Режим доступу: http://www.siegfriedhandschuh.net/pub/2006/whatissemannot2006.pdf Неділя, 11 травня; 2014

[10] Skoutas D, Simitsis A. Ontology-based conceptual design of ETL processes for both structured and semi-structured data. International Journal on Semantic Web and Information Systems. 2007;**3**(4):1-24

[11] Romero O, Abelló A. Automating multidimensional design from ontologies. In: DOLAP. 2007:1-8

[12] Nuseibeh B, Easterbrook S. Requirements engineering: A roadmap. In: Proceedings of the Conference on the Future of Software Engineering. New York, USA: ACM; 2000. pp. 35-46

[13] Jureta I, Mylopoulos J, Faulkner S. Revisiting the Core Ontology and Problem in Requirements Engineering. RE. 2008:71-80

[14] Mazón J-N, Trujillo J. A model driven modernization approach for automatically deriving multidimensional models in data warehouses. ER. 2007:56-71

[15] The ACM Computing Classification Systems [Електроний ресурс] Електрон. текстові дані. Режим доступу: http://portal.acm.org/ccs.cfm?part=author&coll=portal&dl=GUIDE Неділя. 18 травня, 2014

[16] Клинов П. О формальных основах OWL [Електроний ресурс]. Електрон. текстові дані. Режим доступу: http://semanticfuture.net/index.php/ O_формальных_основах_ OWL Четвер. 4 квітня, 2013

[17] Glimm B, Horrocks I, Lutz C, Sattler U. Conjunctive query answering for the description logic SHIQ. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007). 2007;**31**:c. 151-198

[18] Бочаров ВА. Основы логики: підруч. В.А. Бочаров, В.И. Маркин — М.: ИНФРА-М; 2001. 296 с

[19] Noy N, McGuinness D. Ontology Development 101: A Guide to Creating Your First Ontology. Access address: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf, access data: 10.12.2017

[20] Uschold M, Gruninger M. Ontologies: Principles, methods and applications. Knowledge Engineering Review. 1996;**11**(2):93-155

[21] Koval S. Automatic text processing based on the object predicate system, Structural and applied linguistics. St. Petersburg; 1998;(5):199-207

[22] Reisser A, Priebe T. Utilizing semantic web technologies for efficient data lineage and impact analyses in data warehouse environments. In: Proceedings of 20th International Workshop on Database and Expert Systems Application DEXA'09. IEEE; 2009. pp. 59-63

[23] Do HH, Rahm E. On metadata interoperability in data warehouses, technical report. Department of Computer Science, University of Leipzig, Tech. Rep.; 2000. http://doi.uni-leipzig.de/pub/2000-13/

[24] Cui Y, Widom J. Practical lineage tracing in data warehouses. In: ICDE; 2000. pp. 367-378

IntechOpen