Environment. Technology. Resources. Rezekne, Latvia
Proceedings of the 12th International Scientific and Practical Conference. Volume II,28-33

Development of Software for Design Ontological Representations of Production Technologies

Dmitry Andreev
Faculty of Computer Science and Electrical Power Engineering
Pskov State University
Pskov, Russia
dandreev60@mail.ru

Sergey Lyokhin
Faculty of Computer Science and Electrical Power Engineering
Pskov State University
Pskov, Russia
slyokhin@gmail.com

Victor Nikolaev
Faculty of Computer Science and Electrical Power Engineering
Pskov State University
Pskov, Russia
nvv60@mail.ru

Olga Poletaeva
Faculty of Computer Science and Electrical Power Engineering
Pskov State University
Pskov, Russia
oapoletaeva@mail.ru

***Abstract*—The features of the progressive methodological basis for the construction of formalized descriptions of technologies for their subsequent analysis are discussed. The existing possibilities of ontological knowledge engineering applied to the problem of the structural representation of technologies using computer tools are noted. Presented functionality of the developed software that allows automating staged algorithmic procedures for constructing unified decomposition structures of formalized descriptions of technologies. Analysed following development tools: Java Standard Edition programming platform, "Eclipse" IDE, Java programming language, PostgreSQL database management system, "Swing" library for creating a graphical interface and "JGraphX" library for graphs visualization. Database structure of the developed software is described: shown database schema, database tables are defined and the links between them are indicated. The architecture of the developed software is presented: shown data flow scheme and the purpose of each of the modules is described. The main advantages of the developed software "OntoTechnology" is designated, which shows the practical significance of the results.**

Keywords — concept, link, module, table.

I. Introduction

In recent years, methodological conceptions based on ontologies received development in the field of formalization of technological knowledge based on their structural representation. This fact looks logical, since it is ontologies that provide the possibility of comprehensive and detailed formalization of a certain field of knowledge by means of conceptual schemes, which are the systems of concepts interconnected according to certain rules [1] – [3].

The ontology as a generalized scheme of knowledge representation is based on various methods of knowledge conceptualization and methodological considerations of the development of tools for their analysis [4], [5]. Conceptualization is one of the most important processes of human cognitive activity, it implies processing the incoming information and leads to the formation of concepts, conceptual structures and the entire conceptual system in the human mind. Its purpose is constructing an abstract model that determines the structure of the simulated field of knowledge, the properties of its components and causal relations connecting them [6]. Thus, such conceptual model first and foremost consists of certain cognitive structures of special knowledge (mental essences, notions, concepts) and the relations between them.

Technologies need obligatory ontologization before their application, thus, if the knowledge about these technologies goes beyond anthropocentric representations, ontologization becomes the only way of mastering the essence of technologies [7].

The activity of many modern specialists in the field of ontological knowledge engineering varies from the declarative approach while defining possible models of concepts ontologies [8] – [11] and to the implementation of procedural mechanisms for the automated construction of subject fields ontologies [12]. Most of the existing software ontological design tools are aimed at constructing an ontological hierarchy of objects [13], which at best can reflect only the essential aspect of the components of material nature used in the implementation of technologies. In this regard, the usability of such systems, regarding their possible application to display the structural representation of technologies, is significantly limited, since the "coverage

Print ISSN 1691-5402
Online ISSN 2256-070X
http://dx.doi.org/10.17770/etr2019vol2.4064

area" of technological knowledge is, among other things, a set of actions aimed at transforming objects in conditions of specific production.

This article describes the development of software that provides effective operation of technological knowledge through the design of appropriate ontological representations for the subsequent analysis of production technologies.

II. Functionality and the Software Development Tools

For the practical application of the developed method [14], the software was designed to automate staged algorithmic procedures for constructing unified decomposition structures (UDC) of formalized descriptions of technologies.

The developed software has the following functionality:

- Creating a new project, loading a previously saved project, deleting a project that contains a formalized description of the technology;
- Adding, editing, deleting concepts of technological actions, located in the nodes of the decomposition structures of technology (DST);
- Determining the initial degree of content formation of each of the concepts of technological actions;
- Managing the number of private concepts within the UDC based on the formed signs of decomposition;
- Automatic relations establishment between fully formed private concepts within the UDC;
- Automatic determining the complete formation of a holistic concept within the UDC;
- Displaying the results of the design using the library for visualization graphs JGraphX;
- Importing and exporting projects in XML format (eXtensibleMarkupLanguage).

Java Standard Edition (SE) programming platform, "Eclipse" IDE, Java programming language, PostgreSQL database management system (DBMS), "Swing" library for creating a graphical interface and "JGraphX" library for graphs visualization were chosen to develop the software. Currently all these development tools are among the most advanced ones.

Java SE platform is the standard platform Java version 2, designed to create and execute applets and applications intended for individual use and for use in scales of small and medium enterprises [15]. It should be noted that this platform is mainly intended to develop and run desktop applications that do not require preliminary installation on a work computer to start working with them. The advantages of the platform include following features: ability to run applications under the control of most modern operating systems, high reliability and security, portability and high performance.

"Eclipse" IDE is positioned as a free integrated development environment for modular cross-platform applications [16]. It is a fully-fledged Java IDE (Integrated Development Environment) used by a huge community of software developers and it is the corporate standard for application development in many organizations. The main advantage of this design environment is the ability to connect a variety of extensions (modules, plug-ins, etc.) that extend the functionality of the environment for specific practical needs of the developer (for working with databases, application servers, etc.).

"Java" language is distinguished by effective support for the object-oriented programming paradigm [17]. Java programs are translated into a byte code executed by the Java virtual machine (a program that processes byte code and sends instructions to the hardware as an interpreter). The advantage of this method of program execution is the complete independence of the byte code from the operating system and hardware, which allows running Java-applications on any device that has a corresponding virtual machine. Another important feature of the Java language is the flexible security system because the execution of the program is fully controlled by the virtual machine. Any operations that exceed the established permissions of the program (an attempt of unauthorized access to data, connections to another computer, etc.) cause an immediate interruption. The main features of the language are automatic memory management, advanced exception handling, a rich set of I/O filtering tools, a wide range of standard collections, tools for creating multi-threaded applications built into the language and unified access to databases.

PostgreSQL is a free object-relational DBMS [18]. The strengths of PostgreSQL are: support for databases of virtually unlimited size, powerful and reliable transaction and replication mechanisms, an extensible system of embedded programming languages, inheritance and easy extensibility. It should be noted that the use of the ORM (Object-Relational Mapping) method was preferable for working directly with the database. It is a programming technology that allows associating databases with concepts of object-oriented programming languages such as Java, and work with database tables as classes, and with records in tables as objects. This approach allows avoid binding an application to a specific database, but instead, using the application for various database solutions.

"Swing" is a powerful library of graphical components (buttons, input fields, tables, etc.) for creating an advanced user interface. "Swing" refers to the Java Foundation Classes (JFC), which is a set of libraries for developing graphical shells [19]. "Swing" library components support specific dynamically-connected views and behaviours that make it possible to adapt to the graphical interface of the platform, i.e. to the component you can dynamically connect another one, which is specific to the operating system, including the type and behaviour created by the programmer. Therefore, applications that use the Swing library look like native applications for this operating system. Thus, the positive side of such components is the versatility of the interface of the created applications on all platforms.

"JGraphX" is a freely distributed library written in Java and fully compatible with "Swing", which provides the mathematical apparatus of the graph theory [20]. This library is designed to visualize various representations of entities and their relations, including undirected graphs, oriented graphs, subgraphs, multigraphs, graphs with

parallel arcs, etc. The main advantages of JGraphX include the fact that the library allows using different vertex positioning algorithms, as well as creating graphs based on widely used formats, such as XML-documents.

III. DATABASE STRUCTURE

During software development stage tables that comprise the required database were designed. The database schema of the software has the form shown in "Fig. 1".

The database has a relational structure, so data on entities of a formalized description of the technologies are stored in the form of tables consisting of rows (records) and columns (fields). The concept of the primary key (PK), which is a set of fields that uniquely defines a record was also used in the implementation of the design process.

The design of the tables that make up the database of the software was conducted in two stages:

1) Object and connected tables were constructed to describe real entities and their relations with each other;

2) The decomposition of the obtained tables was carried out in accordance with the rules of normalization. As a result, each table began to correspond to three normal forms: 1NF, 2NF and 3NF [21].

To link most of the tables of the designed database, one-to-many relations were used, each of which was represented graphically as a line with symbols at opposite ends "1" и "∞". In cases where two tables were in the potential many-to-many relation, in order to preserve the integrity of the data, link tables were created. Such link tables mainly consisted of two tables' records identifiers only.

Examples of such kind of tables in the presented database schema are:

- "Links_concepts_preceding_ concepts";

"Resulting_components_private_ concepts";

- "Links_concepts_source_ components";
- "Links_concepts_invariant_ components";
- "Values_cost_characteristics".

The database structure of the software is determined by the following tables.

Table I stores information about projects that contain formalized descriptions of technologies. Each record in this table contains information about one project, which is one formalized description of a certain technology.

TABLE I. PROJECTS

Field name	Data type	Description
project_id (PK)	bigint	project identifier
project_name	character vary-ing(255)	name of project
root_concept_id	bigint	identifier of the root concept of the DST

Table II stores information about concepts that are located in the nodes of the DST. Each record in this table contains information about one of these concepts.

TABLE II. CONCEPTS

Field name	Data type	Description
concept_id (PK)	bigint	concept identifier
concept_name	character vary-ing(255)	name of the concept
level	integer	the level of the DST to which the concept belongs
number	integer	an index that uniquely deter-mines the place of the concept in the DST
own_characteristic	character vary-ing(255)	constant character-istic peculiar to the concept
fully_formed	boolean	a logical flag designed to capture the fact that the concept is fully formed
resulting_ compo-nent_id	bigint	identifier of the re-sulting component of the concept
holistic_concept_id	bigint	identifier of the holistic concept for this concept
project_id	bigint	project identifier

Table III stores information about the components of concepts that are located in the nodes of the DST. In this case "components" mean resulting components of the concepts, and their source components. Each record in this table contains information about one certain component.

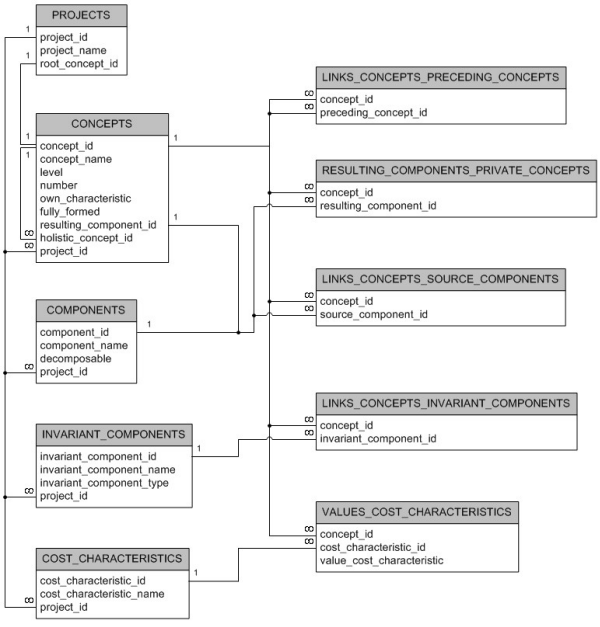

Fig. 1. The database schema of the software

TABLE III. COMPONENTS

Field name	Data type	Description
component_id (PK)	bigint	component identifier
component_name	character_varying(255)	name of the component
decomposable	boolean	a logical flag designed to capture the fact of the presentation of a component of one concept as a set of components of other concepts
project_id	bigint	project identifier

Table IV stores information about the invariant components of concepts that are located in the nodes of the DST. Each record in this table contains information about one certain invariant component.

TABLE IV. INVARIANT_COMPONENTS

Field name	Data type	Description
invariant_component_id (PK)	bigint	identifier of the invariant component
invariant_component_name	character_varying(255)	name of the invariant component
invariant_component_type	character_varying(255)	type of invariant component
project_id	bigint	project identifier

Table V stores information about the cost characteristics of the concepts located in the nodes of the DST. Each record in this table contains information about one cost characteristic.

TABLE V. COST_CHARACTERISTICS

Field name	Data type	Description
cost_characteristic_id (PK)	bigint	identifier of the cost characteristic
cost_characteristic_name	character_varying(255)	name of cost characteristic
project_id	bigint	project identifier

Table VI stores information about the links of private concepts of the same level of decomposition, which is identified with the establishment of the relation of immediate precedence between them. Each record in this table contains information about one binary link of private concepts of the same level of decomposition.

links_concepts_preceding_concepts

Field name	Data type	Description
concept_id (PK)	bigint	identifier of the private concept
preceding_concept_id (PK)	bigint	identifier of the preceding private concept

Table VII stores information about links of private concepts of the same level of decomposition with their resulting components. Each record in this table contains information about one such binary link.

TABLE VI. RESULTING_COMPONENTS_PRIVATE_CONCEPTS

Field name	Data type	Description
concept_id (PK)	bigint	identifier of the private concept
resulting_component_id (PK)	bigint	identifier of the resulting component of the private concept

Table VIII stores information about the links of private concepts of a certain level of decomposition with their source components. Each record in this table contains information about one such binary link.

TABLE VII. LINKS_CONCEPTS_SOURCE_COMPONENTS

Field name	Data type	Description
concept_id (PK)	bigint	identifier of the private concept
source_component_id (PK)	bigint	identifier of the source component of the private concept

Table IX stores information about the links of private concepts of a certain level of decomposition with their invariant components. Each record in this table contains information about one such binary link.

TABLE VIII. LINKS_CONCEPTS_INVARIANT_COMPONENTS

Field name	Data type	Description
concept_id (PK)	bigint	identifier of the private concept
invariant_component_id (PK)	bigint	identifier of the invariant component of the private concept

Table X stores information about the links of private concepts of a certain level of decomposition with their cost characteristics, which allows taking into account specific numerical values of these cost characteristics. Each record in this table contains information about one such binary link additionally with the information about one numeric value of the corresponding cost characteristic.

TABLE IX. VALUES_COST_CHARACTERISTICS

Field name	Data type	Description
concept_id (PK)	bigint	identifier of the private concept
cost_characteristic_id (PK)	bigint	identifier of the cost characteristic of the private concept
value_cost_characteristic	integer	numerical value of the cost characteristic

THE ARCHITECTURE OF THE SOFTWARE

In accordance with the tasks that the software solves, its modular structure and the data flow scheme presented in “Fig. 2”.

The architecture of the developed software is determined by the following modules.

1) The user interface module. This module allows

user managing the software by means of graphical components in the form of buttons located in the right part of the working window of the program; it allows user to enter, edit and delete data; getting information about the structure of the designed formalized description of a certain technology in two modes of viewing: the main view of each of the UDC and the general view of the entire DST.

2) The dialog module. This module is responsible for displaying auxiliary dialog boxes and for implementing mechanisms for their program interaction with the main user form.

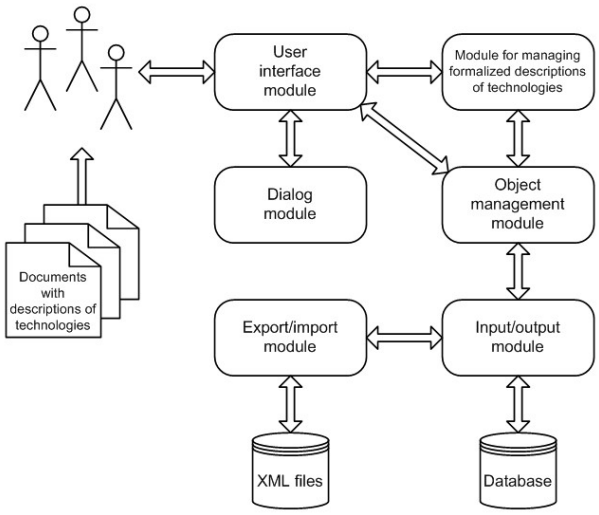

Fig. 2. Data flow scheme

3) The module for managing formalized descriptions of technologies. This module determines the implementation of the basic algorithms of management (formation, structuring and processing) by ontological representations of technologies including all algorithms that implement the method for constructing formalized description of technologies.

4) The object management module. This module allows working with the entities of the ontological representations of technologies as with objects, and allows user to abstract from the structure of storage of the corresponding information in the database.

5) The input/output module. This module is responsible for saving all information about the created or modified formalized descriptions of technologies in corresponding tables of the database. It also allows loading necessary ontological representations of technologies and all related information from the database.

6) Export/import module. This module provides export, i.e. the functionality to save formalized descriptions of technologies in the widely known and practical format of the XML markup language. This allows user to work with the created ontological representations of technologies in other software products that support this format. This module also provides import, i.e. downloading, formalized descriptions of technologies, which are prepared either by the same software, but installed on other workstations, or by other software products that support XML documents format. So, this export/import module provides universal portability of the received design results.

IV. CONCLUSION

The developed software "Ontotechnology" [22], implements new possibilities related to the automation of the process of construction of formalized descriptions of technologies. It supports the procedure for constructing ontological representations of technologies by direct participation of the expert in determining the initial degree of content formation of each of the concepts of technological actions with an explicit indication of their location in the nodes of the DST. The proposed solution allows:

a) Improving the stage of design-technological preparation of production in the part of concentration of processes of information processing necessary for drawing up of the current technical documentation for the technological processes of the enterprise, within one computer program;

b) Increasing the share of automatic procedures in the construction of ontological representations of technologies in comparison with the existing software analogues of this class of systems;

c) Reducing time costs and the need for labour-intensive manual work to obtain aggregate information on technologies, as well as creating new opportunities for rapid obtaining a necessary set of characteristics of the technologies under consideration;

d) Displaying all stages of construction of formalized description of technologies in the form of visual graphic images and providing portability of design results in the format of XML documents supported by most modern information systems.

REFERENCES

[1] J. F. Sowa, „Conceptual graphs as a universal knowledge representation," International journal computers & mathematics with applications, vol. 23 (2-5), pp. 75–94, 1992.

[2] F. F. Vyakkerev, V. G. Ivanov, B. I. Lipsky, and B. V. Markov, Foundations of ontology. Saint-Petersburg: Saint-Petersburg State University, 1997. (in Russian)

[3] A. F. Tuzovsky, S. V. Chirikov, and V. Z. Yampolsky, Knowledge management systems (methods and technologies). Tomsk: NTL, 2005. (in Russian)

[4] T. R. Gruber, „Toward Principles for the design of ontologies used for knowledge sharing," International journal human-computer studies, vol. 43, pp. 907–928, 1992.

[5] I. P. Norenkov, "Intellectual technologies based on ontologies," Information technologies, No. 1, pp. 17-23, 2010. (in Russian)

[6] E. S. Kubryakova, V. Z. Demyankov, Yu. G. Pankratz, and L. G. Luzina, Short dictionary of cognitive terms. Moscow: Philology Department of Moscow State University named after M. V. Lomonosov, 1997. (in Russian)

[7] S. A. Datsyuk, "Ontologization," 2009. [Online]. Available: http://lit.lib.ru/d/dacjuk_s_a/text_0030.shtml [Accessed: Feb. 18, 2019]. (in Russian)

[8] T. A. Gavrilova and V. F. Khoroshevsky, Knowledge bases of intellectual systems. Saint-Petersburg: Piter, 2000. (in Russian)

[9] A. V. Abramov, "Ontology as a method of describing subject areas," Bulletin of Moscow city pedagogical University, Informatics and informatization of education, No. 7, pp. 204-206, 2006. (in Russian)

[10] V. I. Mezhuyev, "Using ontologies as models of subject areas," Artificial intelligence, No. 4, pp. 4–11, 2009. (in Russian)

[11] N. S. Konstantinova and O. A. Mitrofanova, "Ontologies as systems of knowledge storage," 2008. [Online]. Available: http://window.edu.ru/resource/795/58795/files/68352e2-st08.pdf [Accessed: Feb. 18, 2019]. (in Russian)

[12] I. V. Antonov, Method of automated construction of domain ontology, PhD in Technical Sciences [thesis]. Pskov: PSPI, 2011.

(in Russian)

[13] O. M. Ovdey and G. Yu. Proskudina, "Overview of ontology engineering tools," Electronic libraries, vol. 7, issue 4, 2004. [Online serial]. Available: http://rcdl.ru/doc/2004/paper26.pdf [Accessed: Feb. 18, 2019]. (in Russian)

[14] D. A. Andreev and M. V. Voronov, "Method for constructing an ontology of technological actions," Bulletin of Saratov State Technical University, No. 3 (67), pp. 160–168, 2012. (in Russian)

[15] H. Schildt, Eds., Java. Complete guide. Moscow: Williams, 2012. (in Russian)

[16] D. Carlson, Eclipse Distilled. Moscow: Lori, 2008. (in Russian)

[17] B. Eckel, Eds., Thinking in Java. Saint-Petersburg: Piter, 2009. (in Russian)

[18] G. Smith, PostgreSQL 9.0. High Performance. Birmingham: Packt Publishing, 2010.

[19] I. A. Portyankin, Eds., Swing. Spectacular user interfaces. Moscow: Lori, 2011. (in Russian)

[20] JGraph user manual, JGraph Ltd., 2009.

[21] S. D. Kuznetsov, Eds., Database basics. Moscow: INTUIT; BINOM. Knowledge laboratory, 2007. (in Russian)

[22] D. A. Andreev, "Program for automated construction of a formalized description of the technology of the applied field of knowledge OntoTechnology," Certificate of state registration of computer programs No. 2013660420 the Russian Federation, 5 Nov., 2013. (in Russian)