UDC 004.8
DOI: 10.15587/2312-8372.2019.183201

Bibichkov I.,
Sokol V.,
Shevchenko O.

ONTOLOGICAL APPROACH TO DEVELOPMENT OF WEB-CONTENT GENERATION METHOD

Об'єктом дослідження є процес автоматизованого створення веб-контенту, що базується на інформації, представленій в онтологічному вигляді. Одним із найбільш проблемних місць у веб-розробці є процес створення інтерфейсу користувача. Це пов'язано з тим, що даний процес є комплексним та вимагає витрати найбільшої кількості часу та коштів у порівнянні з іншими етапами розробки.

В ході дослідження було застосовано модель розробки програмного забезпечення, що базується на розробці онтології, а потім програмного застосування для її обробки. Даний підхід називається «Ontology-driven development» (або процес розробки програмного забезпечення (ПЗ), що керований онтологією).

Отримано інтелектуальну модель для представлення елементів веб-ресурсів у вигляді онтології, а також програмне ядро системи для генерації веб-сторінок на основі інформації, що зберігається в онтології. Це пов'язано із необхідністю отримання набору кінцевих елементів інтерфейсу користувача (HTML, CSS, JS елементів), із яких формуються веб-сторінки.

Налагодження адресації між сторінками веб-ресурсу має ряд особливостей, зокрема, для зв'язування адреси кінцевої веб-сторінки із контролером, що відповідає за генерацію її змісту, було запропоновано відповідний підхід. Цей підхід функціонує аналогічно так званому «роутеру», який використовується в класичних системах (наприклад, JSP для Java). Відмінною особливістю даного підходу є те, що вся інформація, із якою формується веб-сторінка, а також її адреса зберігається та отримується із онтології.

Завдяки представленому підходу відбувається спрощення проектування та розробки інтерфейсу користувача для веб-проектів, а в перспективі й для інших додатків (настільних, мобільних та ін.). У порівнянні із класичним методом розробки та проектування інтерфейсу користувача, запропонований підхід підвищить можливість повторного використання вже розроблених елементів інтерфейсу користувача. А також забезпечить створення бази готових рішень для розробника у вигляді корпоративної пам'яті.

Ключові слова: *база знань, інтелектуальна модель, модель управління пам'яттю, інтелектуальний аналіз даних, ontology-driven development, корпоративна пам'ять.*

Received date: 18.06.2019
Accepted date: 09.07.2019
Published date: 31.10.2019

1. Introduction

Nowadays, one of the most pressing tasks is the effective management of information resources. Information resource management includes a number of tasks: storing large amounts of information, expanding the knowledge base, editing existing information, and effectively presenting information to the user.

One of the key characteristics in the development of software applications is its cost, which declares the competitive ability in the market of developed software solutions. This characteristic directly depends on the amount of time that was spent on development. In practice, to reduce the time needed to develop a software solution, the method of reusing ready-made tested code proved to be effective. One of the most costly in terms of time is the design phase of the user interface (UI). The proposed method allows in the long term to reduce the time for developing a software application due to the automatic generation of UI elements based on the information contained in the knowledge base.

Thus, the use of the proposed method and knowledge model not only reduces the time for developing software, and therefore its cost, in addition, increases the possibility of reusing ready-made solutions in the UI design.

The main disadvantage of the presented method is the limitations of its use by systems based on ontological knowledge bases. To use the proposed method, it is necessary to store the ontology separately and as part of the main data warehouse and configure interaction with it.

2. The object of research and its technological audit

The object of research is the process of creating structural elements of the UI for web pages. The main idea is in providing automatic generation of UI elements based on information stored in the knowledge base (KB) in an ontological form. In the framework of this work, on the example of the «Portal of Independent Evaluation of the Quality of Higher Education» [1], the possibility of creating an ontological description of UI elements and generation of UI for web pages from [2] is considered. In particular, structural elements that are described in an ontological form include web page templates, block and structural elements.

Finished web pages are generated using the system software core, developed in the Java programming language based on the ontological description located in the knowledge base (KB) on a dedicated server. The main elements of web pages are described in a hierarchical form.

Within the framework of the developed system, taking into account the peculiarities of ontology construction, three main entities are identified, on the basis of which web pages are described: «Classes», «Properties» and «Instances» (or final objects that are instances of classes). It should be noted that the ontological approach to work with data divides «Properties» into two groups: «Object Property» and «Datatype Properties». Object properties combine class instances with each other. In turn, data type properties combine class instances and target data. Classes and properties are presented in a hierarchical form. The class diagram is shown in Fig. 1.

It should be noted that in Fig. 1 is a diagram of a developed class hierarchy adapted for web page development. This structure can be expanded and adapted to generate UI for a number of other technologies, for example, Android applications (XML interface), WPF (XAML markup), Java FX (FXML markup), and software ontological systems with built-in proactive component [3]. The set of classes governs the possible structural elements on the basis of which the ontological model for creating the UI will be further built.

The ontological model, or ontological representation of a web page, is the basis for the formation of the HTML structure of the document. The core of the system is responsible for interpreting the ontological model into the final web interface of the page.

The hierarchical structure of representing UI elements as a root element has the «Application» class.

The presentation entity class is inherited from the application and combines the main classes of element groups on the basis of which the UI will be built: «execute», «ui_element» and «container».

The «execute» class combines classes to create dynamic UI elements: loops, conditional statements, and the like. For example, on the basis of the «for_each» class, it is possible to create new password elements for displaying lists based on a data collection obtained from the database.

The «ui_element» class combines classes to create basic standard UI elements:

– «resource_tree» used to generate hierarchical structures based on data from the database;

– «label», which is a universal element for displaying data obtained from the database. This class is used to generate elements for displaying headers, resource properties, and the like. The «label» class is also a child of the «value_of», used, in particular, inside the «for_each» loop to access the list of received elements (term) received as a result of the SPARQL query;

– «input» – the class on the basis of which input elements are generated. This class has the «value» property to generate element default values (similar to the «value» attribute in HTML). In addition, «input» has the «caption» property, which allows to generate the corresponding element with a mask for input (similar to the «placeholder» attribute in HTML) and «name» for accessing the entered data and processing it;

– «button» – the class on the basis of which the button elements are generated. This class has three attributes «caption», «value» and «name», which are used to generate the attribute of the button header, and the name «name» pairs are the value «value», respectively, from which information will be sent to server;

– «container» – the class based on which specific page layout elements are generated for the vertical «vertical_container»and horizontal «horizontal_container» layout of the elements. In addition, HTML code is generated based on the «block_container» containers for a number of UI elements combined into a single block. Such corresponding HTML elements can be , , their combinations, etc.

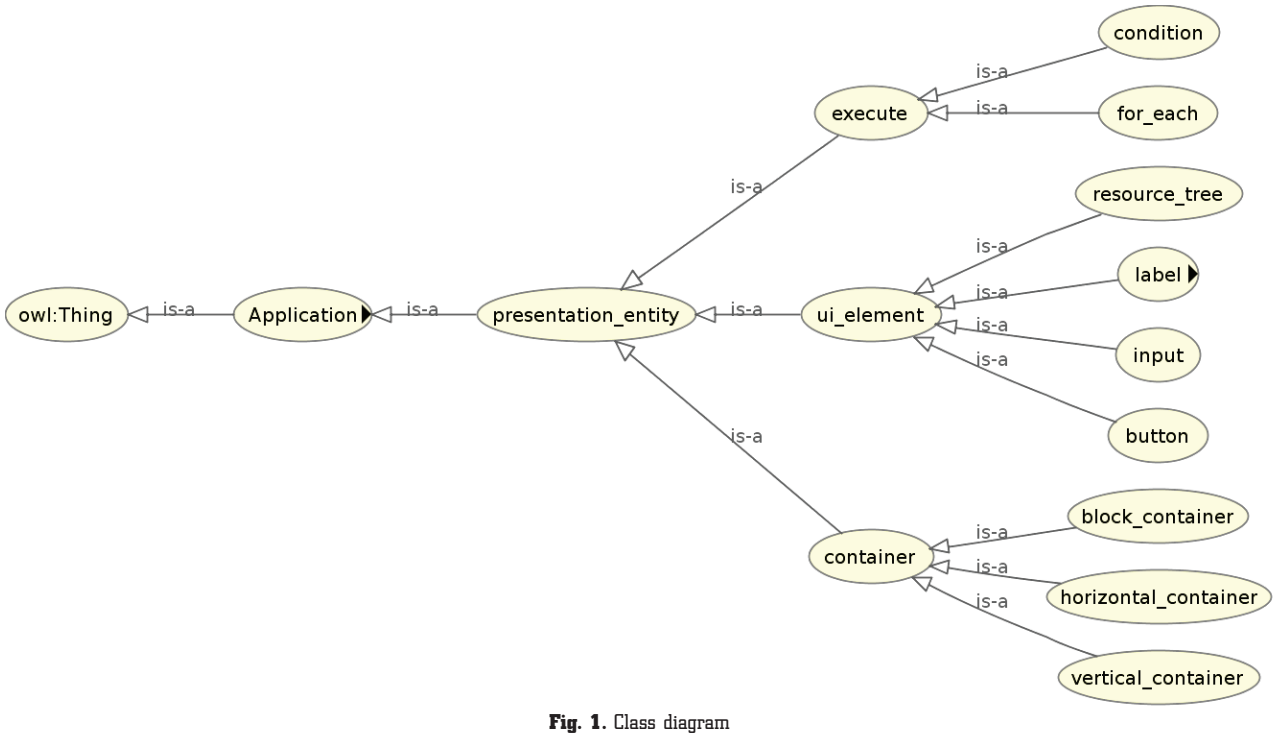

Fig. 1. Class diagram

ISSN 2664-9969

An important aspect is the flexibility and extensibility of the proposed method. So, in order to create a new, additional, for example, block element, there is no need to make structural changes to the existing class hierarchy and the relationships between them, it can initiate the process of structural modifications of the system software core. But it's enough, in case of expanding the UI element base, add a new Instance of an existing class or a new class and, if necessary, new properties to the knowledge base. Also, this system can be modified to work with other types of interfaces. At the same time, there is no need to delete the existing element base; instead, new elements are added to those already present in the KB. Thus, in the case of a change in the subject area, extensions are used instead of the processes of replacement and removal. This allows the use of knowledge base equally for a new subject area, and for an existing one; for various types of applications or applications.

The disadvantages of this method are the increased complexity of the primary design and the relatively large amount of data that must be created. These factors significantly increase the cost of initial system development. There is also a risk of a significant loss in system performance with a significant increase in the ontology volume.

3. The aim and objectives of research

The aim of research is development of an effective method for the automated creation of a user interface based on the ontological form of the structural elements of a web page [4].

To achieve this aim, it is necessary to complete the following objectives:

1. Design and develop an ontological description of the structural elements of the interface.

2. Develop the software core of the system for generating structural elements of web pages based on the information contained in the ontology.

3. Develop a method for addressing web pages within a web resource.

4. Optimize ontology processing processes [5, 6].

4. Research of existing solutions of the problem

Among the main directions of solving the problem of automated development of the user interface, identified in world resources, it is possible to single out [7], but in this proposed approach, the possibility of automated generation of the interface is considered.

In [8, 9], the possibility of creating a user interface for various technologies based on the so-called «Model Driven Development» approach is considered, but the possibility of creating a database of already created UI elements to ensure their reuse was not considered.

The authors of [10] consider a method for automatically generating a user interface based on the use of XML technologies for JavaFX applications, but the possibility of adapting the developed UI to other technologies is not considered.

The possibility of creating a user interface for different platforms, based on the «Model Driven Development» approach, is shown in [11], but the question of the final integration of the output UI into real systems remains open.

An alternative solution to the problem is described in [12], which presents technologies for creating a user interface for the Internet of Things, does not provide for the possibility of forming UI elements for compatible technologies (e. g., GUI for desktop applications).

The author of [13] considers the possibility of creating a human-machine interface by developing a special programming language, but the question remains about the possibility of expanding the existing base of UI elements.

In [14, 15], technologies for forming the user interface are presented, in specific technologies for web and mobile technologies, respectively. However, the process of integrating the created UI into existing systems and the possibility of adapting it to other technologies remains open.

But the work [16] is devoted to the problem of creating a user interface for the main types of software: web applications, mobile devices. But this work does not provide for the possibility of creating a UI for desktop applications, and the process of storing program code for already developed UI elements is not sufficiently disclosed.

Thus, the results of literature analysis allow to conclude that the approach presented combines the strengths of analogues, takes into account their shortcomings and expands the scope of application of the above methods.

5. Methods of research

The so-called ODD (Ontology-Driven Development – a method for developing software applications, which involves first creating an ontology, and then programming code) is chosen as a basis, in the framework of this research [17]. This approach consists in designing and creating an ontology in the first stage of development and writing program code in the second stage. In the study of systems that provide adaptation to changes in the subject area, it is discovered the advantages of using this approach [6].

For the design and development of the software kernel, a stack of technologies based on Java [18] is used, the main of which is Apache Jena [19]. To access the information stored in the KB, the SPARQL query language is used [20]. A universal Virtuoso server was used for storage and processing of knowledge bases [21]. In addition to the main purpose, Virtuoso also provides additional optimization of the processes of interaction with the KB by creating an object model based on a loaded ontology. This model is also used by Jena, which simplifies the process of accessing information stored in the database.

6. Research results

In the course of research, an ontology-based method for generating web content is created, an ontological model is designed and developed to describe the structural elements of the UI for web pages. An example of the page and its ontological description is shown in Fig. 2.

Web page in Fig. 2 consists of the following elements: data input field, horizontal block, in which it is from left to right: hierarchical structure of resources located in the database; list of selected resources; resource description. The ontological description of the web resource is shown in Fig. 3.

Fig. 3 depicts an ontological description of the final web resource, presented in object form. The basic unit used to describe the object is an object, within the ontology has the definition of «Instance».

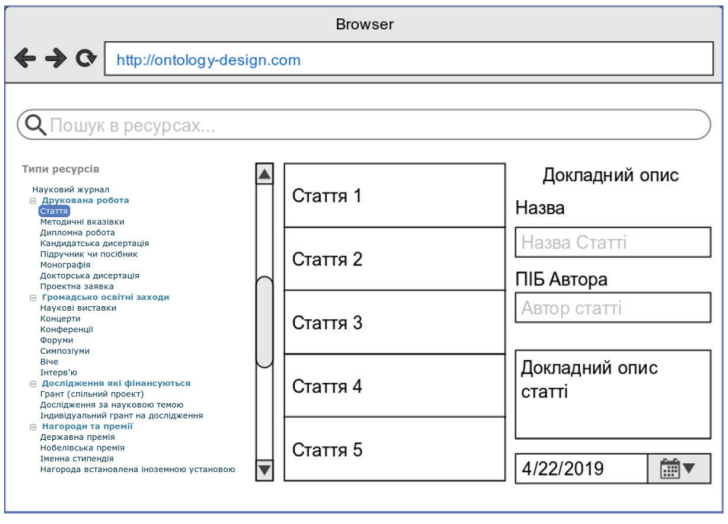

Fig. 2. Example of web page

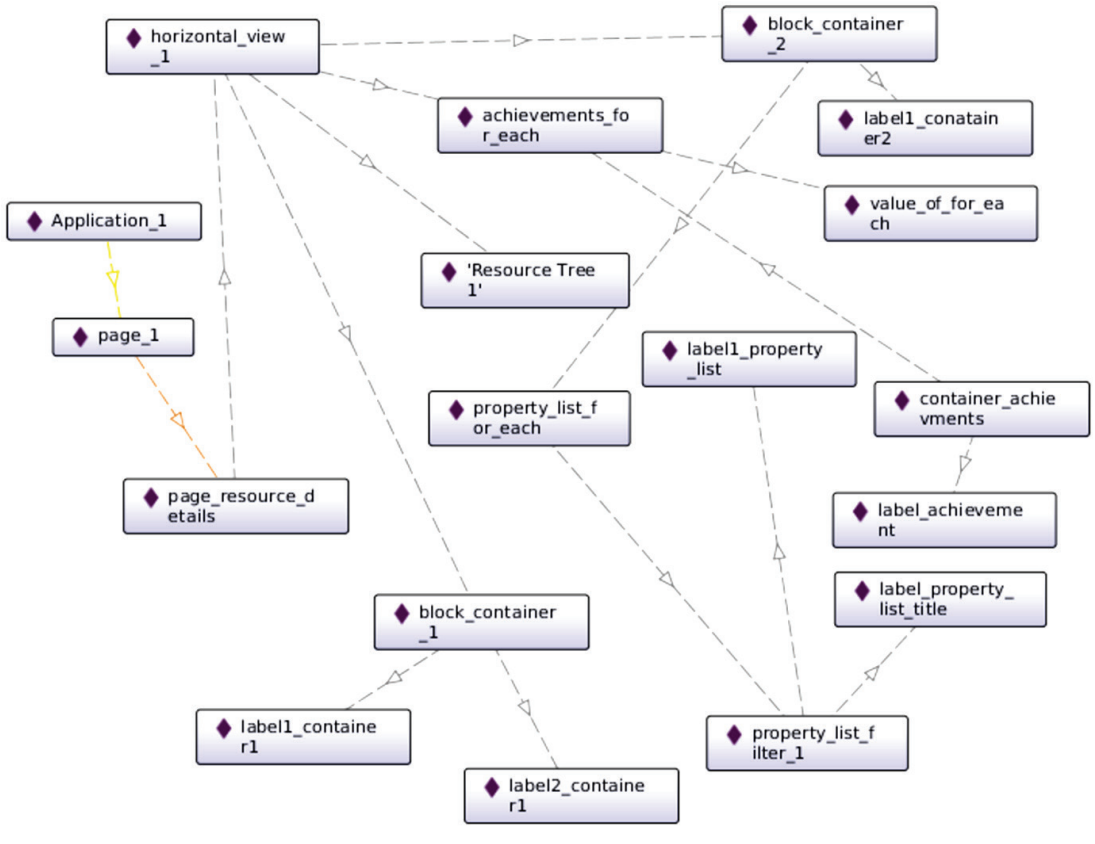

Fig. 3. Ontological description of the web resource

«Instance» is an object built on the basis of a particular class, connected by object properties or data type properties with other objects. Instance is also called a resource. Instance may contain specific data or other instance. In order to place objects in Instance, object properties are used; in order to add data to Instance, use data type properties.

It should be noted that all data in the ontology is typed. Complex data types can be created based on existing ontologies. So, during the development of the ontology, additional data types are created: «elementType», «sparqlCode», «updateFreaquency».

The proposed structure of UI elements is built in the form of a hierarchy, has a root element, siblings and child elements. The root element is «Application_1», which combines all the pages of a web resource.

For the structure of a separate web page, which is shown in Fig. 3, the root element is «page_1». This object contains the address at which the page is placed, as well as the name of the controller that is associated with it, used by the kernel to generate web content with an ontological model. This information is presented in the ontology using the properties of data types.

«Page_resource_details» is an object of the «page_view» class and includes the elements that make up the page. In this case, «page_resource_details» includes a single «horizontal_view_1» element, which in the ontology context

ISSN 2664-9969

means that «page_resource_details» is connected via the «element» property to «horizontal_view_1».

«Horizontal_view_1» is an object of the «horizontal_container» class, which inherits from the «container» class and determines the location of UI elements on the page. «horizontal_view_1» represents the construction of the page in the form of structural elements located horizontally relative to each other. This object is connected through the object «element» property with «block_container_1», «Resource Tree 1», «achievements_for_each» and «block_container_2».

«Resource Tree 1» is the object of the «resource_tree» class, which is inherited from the «ui_element» class. This element is used to describe the hierarchical data structure that will be built on the page.

«Block_container_1» and «block_container_2» are objects of the «block_container» class that inherit from the container class and represent a description of the block elements of a web page (for example, «div»).

Container elements (that is, class objects that inherit from the «container» class or its descendant classes) can include simple elements from which the UI will be generated – these are class objects: «label», «button», «input», and also objects used to dynamically create UI elements: «for_each», «condition». The object classes of the first group of elements are descendants of «ui_element». The object classes of the second group are inherited from the execute class.

The objects of the «execute» class are used to dynamically generate UI elements. Based on the data labeled in these elements, the software core can generate web page elements for displaying data from the knowledge base: «achievements», «property_list». These elements correspond to the HTML «select» and «option» structural elements, respectively.

A complete list of elements contained in the ontology with the indicated relationships between them (Object Property, Datatype Property, Datatype) is shown in Fig. 4.

Fig. 4 shows the structure of the ontology with the indicated properties connecting the objects of the classes. In addition, Fig. 4 shows the data types pointed to by the properties of data types.

From the point of view of economic profitability, the use of this method will reduce in the long run the time spent on designing and developing the UI and, as a result, will reduce the cost of development. In addition, the use of this approach in the future will increase the chances of unifying the UI for web resources and facilitate the migration of the finished UI for other technologies, such as «WPF», «fxml», etc.

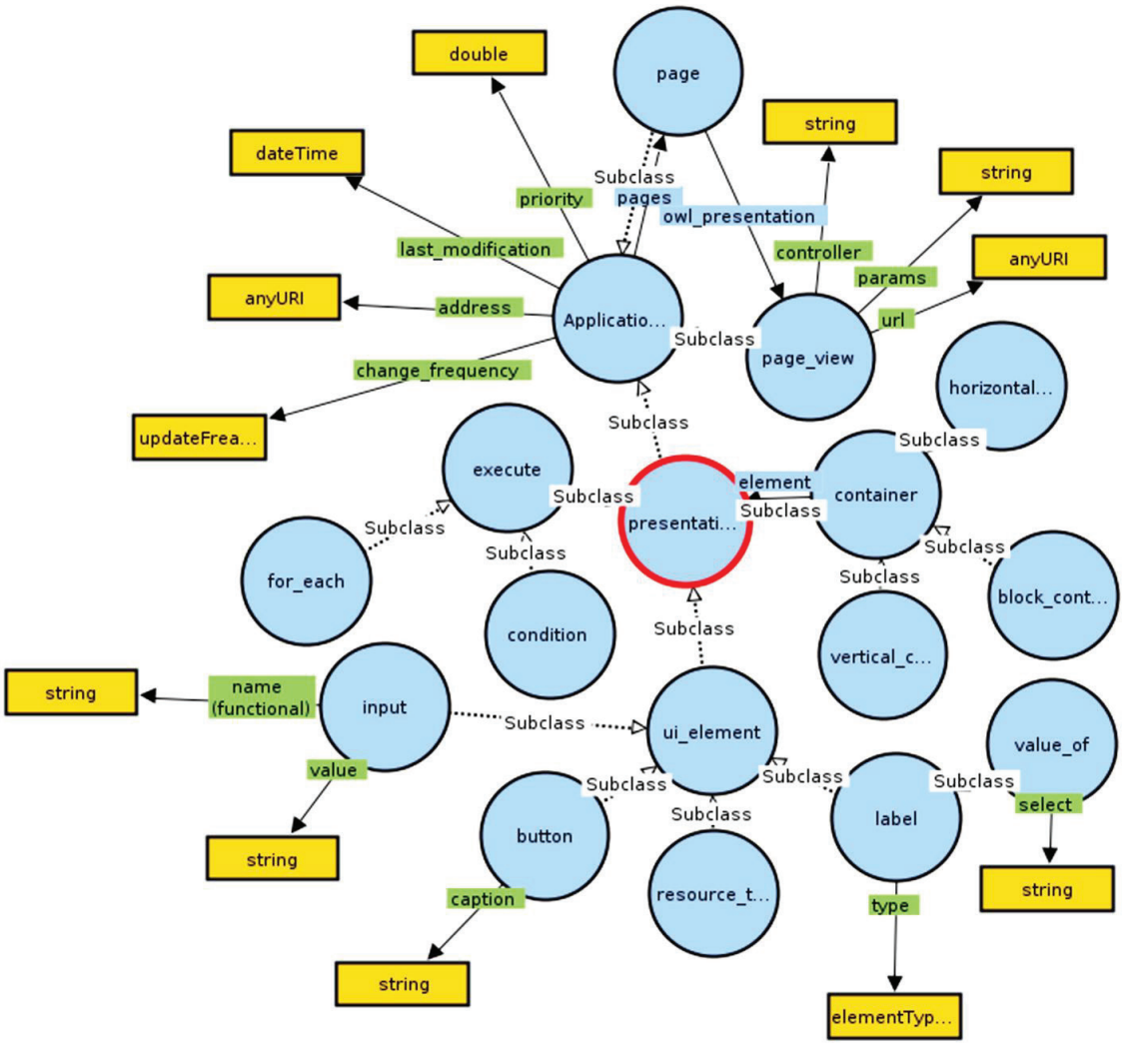

Fig. 4. Schematic representation of an ontology with properties

7. SWOT analysis of research results

Strengths. Using this method, in the long run, will reduce the cost of designing and developing UI for projects of various kinds: web resources, desktop applications, applications for mobile devices, etc. This is possible by unifying the description of already created UI elements, forming a base from them, may expand or change; at the same time, the effect of changes on existing elements in the knowledge base is minimized.

Compared with the analogues of this system, the cost of development is reduced and the productivity of UI development as a whole increases.

Weaknesses. The weaknesses of this method include the necessary initial time spent on designing the elements on the basis of which the UI will be built. In addition, the use of this approach will increase the complexity of the development and increase its initial complexity.

Opportunities. The development prospects of this method include expanding an existing set of descriptions of UI elements, simplifying ontology design and adding data to it, by developing a specific software application for the automated generation of an ontological description of elements according to its description in a natural language of communication. Using the proposed approach at the enterprise will allow end developers to create a common space in the form of corporate memory for storing the created descriptions of UI elements and the practice of their implementation in ready-made solutions.

Threats. The threats of the main threats associated with the use of this method include the risk of using methods that are easier to implement at the enterprise. In addition, to support the design of the knowledge base, which will contain a description of the elements of the UI, it is necessary to have a specialist who is qualified in the use of ontologies. In this regard, it may be necessary to hire an additional employee at the enterprise, or invest in the development of specialists, already at the enterprise, to obtain the necessary knowledge and qualifications in this area.

8. Conclusions

1. The ontological description of the structural elements of the interface is designed and developed. The result is a consistent ontology for storing elements of the developed interface. The developed ontology consists initially of 17 base classes, 15 properties and 18 objects. It also provides for the possibility of expansion for the integration of UI elements of related technologies (mobile solutions, desktop applications) and addition of UI elements for creating web applications.

2. The expanded software core of the system [1] to provide the possibility of generating structural elements of web pages based on the information contained in the ontology. As a result, additional classes and methods of the system software core were developed that are integrated into the existing system architecture.

3. A method for addressing web pages within a web resource is developed. As a result of this, the main ontology is expanded to provide storage for internal links. The software core of the class hierarchy system is also expanded with methods for organizing addressing between pages within a web resource.

4. After carrying out the necessary modifications to the system core and creating and integrating an additional ontology, a 5–10 % loss in system performance is detected. To restore the system's speed, a set of methods for optimizing the work are described [22]. As a result, an increase in system performance by an average of 10–15 % is obtained.

References

1. *TRUST Portal*. Available at: http://portal.dovira.eu/ Last accessed: 17.10.2019
2. Terziyan, V., Golovianko, M., Shevchenko, O. (2014). Semantic Portal as a Tool for Structural Reform of the Ukrainian Educational System. *Information Technology for Development, 21 (3),* 381–402. doi: http://doi.org/10.1080/02681102.2014.899955
3. Terziyan, V., Shevchenko, O., Golovianko, M. (2014). An introduction to knowledge computing. *Eastern-European Journal of Enterprise Technologies, 1 (2 (67)),* 27–40. doi: http://doi.org/10.15587/1729-4061.2014.21830
4. Shevchenko, A. Iu., Shevchenko, E. L. (2011). How to bring artificial intelligence into the Clouds. *Eastern-European Journal of Enterprise Technologies, 3 (12 (51)),* 66–70. Available at: http://journals.uran.ua/eejet/article/view/2472
5. Bibichkov, I. E., Sokol, V. V., Shevchenko, A. Iu. (2014). Optimizing the performance of ontological knowledge bases built on the basis of «VIRTUOSO». *Eastern-European Journal of Enterprise Technologies, 5 (2 (71)),* 4–8. doi: http://doi.org/10.15587/1729-4061.2014.28553
6. Shevchenko, A. Y., Shevchenko, E. L. (2012). Modern ontological database management systems comparison. *Visnik SEVNTU, 131,* 82–86.
7. Offenhartz, J. K., Dana, D. (2017). *Dynamic generated WEB UI for configuration*. published: 17.04.17.
8. Bernaschina, C., Comai, S., Fraternali, P. (2017). Online Model Editing, Simulation and Code Generation for Web and Mobile Applications. *2017 IEEE/ACM 9th International Workshop on Modelling in Software Engineering (MiSE),* 33–39. doi: http://doi.org/10.1109/mise.2017.1
9. Roubi, S., Erramdani, M., Mbarki., S. (2016). A Model Driven Approach for generating Graphical User Interface for MVC Rich Internet Application. *Computer and Information Science, 9 (2),* 91. doi: http://doi.org/10.5539/cis.v9n2p91
10. Yongzhi, Y., Peng, Z., Yun, X. (2017). Automatic User Interface Generating for Simple Interaction in Pervasive Computing. *22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2,* 3–8. doi: http://doi.org/10.1109/cse-euc.2017.187
11. Bouraoui, A., Gharbi, I. (2019). Model driven engineering of accessible and multi-platform graphical user interfaces by parameterized model transformations. *Science of Computer Programming, 172,* 63–101. doi: http://doi.org/10.1016/j.scico.2018.11.002
12. Johnsson, B. A., Magnusson, B. (2017). Towards end-user development of graphical user interfaces for internet of things. *Future Generation Computer Systems*. doi: http://doi.org/10.1016/j.future.2017.09.068
13. Gaouar, L., Benamar, A., Le Goaer, O., Biennier, F. (2018). HCIDL: Human-computer interface description language for multi-target, multimodal, plastic user interfaces. *Future Computing and Informatics Journal, 3 (1),* 110–130. doi: http://doi.org/10.1016/j.fcij.2018.02.001
14. Taivalsaari, A., Mikkonen, T., Systä, K., Pautasso, C. (2018). Web User Interface Implementation Technologies: An Underview. *Proceedings of the 14th International Conference on Web Information Systems and Technologies, 1,* 127–136. doi: http://doi.org/10.5220/0006885401270136
15. Yannes, Z., Tyson, G. (2019). Amniote: A User Space Interface to the Android Runtime. *Proceedings of the 14th International Conference on Evaluation of Novel Approaches to Software Engineering, 1,* 59–67. doi: http://doi.org/10.5220/0007715400590067
16. Engel, J., Märtin, C., Forbrig, P.; Kurosu, M. (Ed.) (2017). Practical Aspects of Pattern-Supported Model-Driven User Interface Generation. Human-Computer Interaction. User Interface Design, *Development and Multimodality. Cham: Springer International Publishing,* 397–414. doi: http://doi.org/10.1007/978-3-319-58071-5_30

ISSN 2664-9969

17. Gharbi, G., Ben Alaya, M., Diop, C., Exposito, E. (2012). AODA: An autonomic and ontology-driven architecture for service-oriented and event-driven systems. *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. WETICE, 3,* 72–77. doi: http://doi.org/10.1109/wetice.2012.84
18. *The Java™ Tutorials*. Available at: https://docs.oracle.com/javase/tutorial/ Last accessed: 24.03.2019
19. *Apache Jena – Jena tutorials*. Available at: https://jena.apache.org/tutorials/index.html Last accessed: 24.03.2019
20. Zhang, Y. (2015). Research on Efficient SPARQL Query Processing for RDF Data. *Proceedings of the 2015 2nd International Workshop on Materials Engineering and Computer Sciences*, 476–482. doi: http://doi.org/10.2991/iwmecs-15.2015.94
21. *OpenLink Virtuoso Universal Server: Documentation* (2007). Available at: http://docs.openlinksw.com/virtuoso/
22. Bibichkov, I., Sokol, V., Shevchenko, O. (2017). Ontological knowledge bases productivity optimization through the use of reasoner combination. *Eastern-European Journal of Enterprise Technologies, 5 (2 (89))*, 49–54. doi: http://doi.org/10.15587/1729-4061.2017.112347

Bibichkov Igor, *Assistant, Department of Artificial Intelligence, Kharkiv National University of Radio Electronics, Ukraine, ORCID: http://orcid.org/0000-0003-1424-6960, e-mail: bibi4kov@gmail.com*

Sokol Vadym, *Postgraduate Student, Department of Artificial Intelligence, Kharkiv National University of Radio Electronics, Ukraine, ORCID: http://orcid.org/0000-0003-2461-3453, e-mail: sokol@sw-expert.com*

Shevchenko Oleksandr, *PhD, Associate Professor, Department of Artificial Intelligence, Kharkiv National University of Radio Electronics, Ukraine, ORCID: http://orcid.org/0000-0002-0068-4698, e-mail: shevchenko@sw-expert.com*

UDC 004.891
DOI: 10.15587/2312-8372.2019.183301

Dabahian D.

DETERMINATION OF INTELLECTUAL ACTIVITY IN SOLVING THE PROBLEMS OF BANK FUNCTIONING OPTIMIZATION

Об'єктом дослідження є процес функціонування комерційного банка. Одним з найбільш проблемних місць є оптимізація роботи банка згідно з вимогами клієнтів в умовах обмежених ресурсів, тобто, як потрібно розподілити певну суму інвестицій по різним напрямкам діяльності банка оптимальним чином. Під цим слід розуміти максимальну задоволеність клієнтів процесом функціонування банка. Також слід визначати важливість напрямків діяльності в залежності від зворотнього зв'язку з клієнтами – збираючи клієнтську інформацію, таку як скарги, побажання, результати опитів та інше. Такі дані не є інтелектуальними, їх треба формалізувати та на цій основі побудувати стратегію функціонування банка на певний період часу.

В ході дослідження було використано висхідний підхід до створення систем штучного інтелекта. На основі не інтелектуальних даних (підсистеми банка, клієнтські дані, статистика) визначається інформація для побудови інтелектуальної діяльності стосовно прийняття рішень щодо оптимізації процеса функціонування банка в цілому, як єдиної системи, тобто побудови оптимальної стратегії діяльності банка.

В результаті дослідження отримано проект інтелектуальної системи, яку призначено для побудови оптимальної стратегії діяльності в умовах обмежених ресурсів. Для заповнення бази знань цієї системи проведено структурування та формалізацію знань. Оптимальним варіантом для цього дослідження було визнано формально-логічну модель на основі побудови предикатів першого порядку.

Завдяки цьому забезпечується можливість реалізації інтелектуальної системи для вирішення проблеми розподілу внутрішніх інвестицій банка оптимальним шляхом, тобто, з максимальним підвищенням рівня задоволеності клієнтів. Використання цієї системи на практиці має допомогти керівництву банка розподілити певний обсяг внутрішніх інвестицій у напрямках діяльності банка оптимальним шляхом, керуючись побажаннями клієнтів.

Ключові слова: *ефективні відносини «банк-клієнт», інтелектуальна система, оптимізація функціонування банка, розподіл інвестицій.*

Received date: 11.06.2019
Accepted date: 03.07.2019
Published date: 31.10.2019

1. Introduction

To date, artificial intelligence systems are already used in banking [1–3]. This allows to save time manual processing of some banking information, to reduce the amount of the same type of work that bank employees perform and so on.

Standard financial products and services designed for a wide range of consumers – this is yesterday. The modern client needs the personified conditions for deposits, loans and other offers. To realize this without an individual approach is impossible. Here, the banks come to the aid of artificial intelligence.