Date of publication SEP-30, 2023, date of current version AUG-23, 2023.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.22.3.3234

Approach to Implementation of Configuration Process for Adaptive Software Systems based on Ontologies

DMYTRO FEDASYUK, ILLIA LUTSYK

1Department of Software, Lviv Polytechnic National University, Lviv, 79000, Ukraine
(email: dmytro.v.fedasyuk@lpnu.ua, illia.i.lutsyk@lpnu.ua)

Corresponding author: Illia Lutsyk (e-mail: illia.i.lutsyk@lpnu.ua).

ABSTRACT Analysis of scientific research on the development of adaptive and self-adaptive software systems is conducted. It is established that the use of machine learning methods and feedback diagrams is an effective way to design and develop adaptive software. It is determined that the existing methods do not fully provide the possibility of dynamic changes and expansion of functional and graphic characteristics. The software adaptation process is designed based on the ontological model using the semantic decision-making mechanism. The proposed method allows us to dynamically determine the necessary system characteristics and perform software adaptation. Modification process takes into account the information about currently active device based on data about the needs and requirements of the user. Using the results of designing an abstract approach to software configuration modification, an experimental study of the speed of generating optimal system settings is conducted. According to the results of the experiment, it is established that the new method demonstrates 20% better indicators of the speed of generating software settings compared to classical approaches.

KEYWORDS adaptive software systems; ontological approach; onto-oriented systems; methods of software adaptation; configuration of software systems.

I. INTRODUCTION

RAPID development of new technologies for the development and implementation of software entails an increase in the number of requirements for software systems. Such growth affects the overall structure and content of the software, since each new requirement requires a change in the functionality or graphical component of the system. As a result, the problem of effective configuration of individual system modules arises, since the set of software components may vary depending on user requirements.

The basic requirement of modern software systems is to solve the tasks of operational support of decision-making based on the operation of large arrays of information. For the effective operation of such tools, it is necessary to adapt them to the specifics of a particular problem area and options for user requirements [1, 2]. In such cases, it is advisable to use artificial intelligence. Automation of decision-making in such systems requires certain settings, which are based, as a rule, on expert refinement of coefficients or formation of knowledge bases for training samples of neural networks.

II. PROBLEM STATEMENT

The knowledge base of adaptive systems needs a tool that would allow selecting relevant options for software configuration implementation not only at the installation stage, but also when there is a need to change the requirements during the functioning of the software product. Thus, there is a problem of developing a toolkit that will allow faster execution of the tasks of creating and modifying components of the software architecture. In addition, a low level of coupling will avoid many problems associated with adapting the system to the requirements of a specific user.

It should be noted that in order to ensure the ability to dynamically change the characteristics of the created software solution, two types of systems are usually used: adaptive and self-adaptive. The difference between these two approaches lies in the process of determining the need to modify the characteristics and properties of software solutions. Unlike adaptive systems, self-adaptive systems are able to independently analyze specific characteristics and subsequently change their behavior depending on the received assessment of efficiency or productivity [1].

In addition, the problem of software adaptation consists in the effective formation of a stable system configuration that would be optimal for a certain user or group of users and would not require significant system resources. For this, during the development of adaptive systems, external and additional components should be defined with a sufficient level of abstraction to reduce the degree of coupling and improve the efficiency of the system.

Therefore, taking into account the indicated problems and challenges, it is relevant to design the configuration modification process for adaptive and self-adaptive software systems, which would allow improving the software modification in accordance with changing requirements and user preferences taking into account the minimization of the use of system resources.

III. RELATED WORK

In a number of studies, general principles and methods of designing and implementing self-adaptive systems have been defined. In particular, in [1, 3] the authors highlighted the problems of designing these systems, namely the problems of defining new unified processes of software adaptation and decentralization of system elements. Based on the taxonomy of approaches, the authors identified new directions for the development of the process of adapting software systems:

- contextual adaptation – activators of managed resources are integrated into the process of forming judgments;
- decentralization of adaptation logic – involves the use of a decentralized management system;
- proactive adaptation – software adaptation is based on prediction.

The indicated directions and problems of the implementation of software adaptation are solved by applying one or more adaptation methods [4]:

- Approach based on feedback loops (feedback loop) - adaptation takes place on the basis of constant monitoring and control of the system, which allows making changes depending on the context [5, 6];
- Learning-based method – the method consists in using the principles of machine learning. The process of adaptation of the decision support system can be achieved effectively by configuring the system for dynamic learning and the corresponding change of components [7-9];
- Planning-based method – planning methods for adaptive systems determine the priority and order of adaptation for one or more elements of the system. Such methods are divided into those, in which only one or two changes at a time are taken into account and the elements are assumed to be independent with unchanged priority, or search-based optimization methods, which are more effective for a larger number of variable factors [10-12];
- Method based on monitoring – the approach is applied in several stages. The application is first traced in a simplified form, providing information for the next stage, which determines the adaptive configuration and selects traces according to the current configuration. Adaptive configuration is determined by a set of criteria specified through the proposed subject-oriented language [13, 14];

It should be noted that despite the general effectiveness and diversity of software adaptation methods, there is a problem of domain representation. For adaptive and self-adaptive systems, the elements of the subject area can be changed, supplemented or deleted, which makes it impossible to predict in advance the parameters for the effective operation of the system.

The concept of creating context-adaptive user interfaces for commercial vehicles was revealed in the work of L. Schelkopf, M. Wolf [15]. The authors noted that the use of context-adaptive interfaces makes it possible to detect the most recurring events. Based on the received information, it becomes possible to build different variations of the user interface and scenarios of the system use for the problem area analyzed by them, defining the main "working phases" of commercial transport drivers.

Another example of the implementation of adaptive user interfaces based on conceptual and ontological models was presented in the work of D. S. Nehurytsya, E. V. Sokolova [16]. The article defined the components of Web-oriented systems based on the analysis of methods of building and ensuring the quality of user interfaces. In addition, the authors considered the features of adaptive interfaces, which made it possible to define the concept of the user model as a mandatory component of adaptive software interfaces of Web-oriented systems. As a result, the paper presented an empirical study that proves the need to define additional metrics for evaluating search quality and classifying user actions and interests in a session of working with the Web system.

A promising way to describe subject areas is a combination of the principles of designing and developing ontological models and the principles of fuzzy logic [17, 18]. The authors pointed out that the design of ontologies during the creation of complex software usually affects the adequacy of the model and causes inconsistency in the development process. For this, it is advisable to use a combination of the ontological approach and elements of fuzzy logic. As a result, the authors showed that the modified model of ontology improvement in time allows effectively solving the problems of analysis and evaluation of the state space of evolving ontologies using elements of fuzzy logic.

In addition to the effective representation of the subject area for the implementation of an adaptive software system, architecture is also an important component. Depending on the context, type and other adaptation processes, the architecture should provide a sufficiently high level of abstraction that would not allow one to be tied to a specific implementation.

A correct definition of the architecture is especially necessary for distributed systems, which are often denoted as a system of systems (SoS) [19, 20]. SoS are built from a potentially large number of subsystems, which consist of various components, services and resources. In addition, SoS are usually deployed in environments where the context changes, which in turn causes the need to deviate from the pre-designed workflow and form a more complex system. Such modification, in turn, allows us to maintain the intended abstract behavior or be able to provide new behavior that is possible only through integration with other systems in a new context.

A number of works are aimed at effective implementation of the SoS concept and architecture. In particular, work [21] presented an effective solution for architectural design according to the SoS concept for Industry 4.0 in the field of construction. The authors emphasized that the development of complex systems requires constant interaction between design decisions at the SoS level and decisions at the level of its constituent systems, which require adaptation as the SoS develops. In the article, the authors developed a description of the SoS architecture based on ISO 42010, which consisted of representations related to hierarchy, asset integration, communication, information. However, despite the effective description of architecture for a specific domain, the authors noted that there was a lack of architectural frameworks suitable for this.

In addition to using standard methods of describing SoS, it is also effective to combine traditional approaches with ontological ones. Thus, in [22], the process of designing the architecture for complex software systems based on ontology was reflected. The authors proposed a generalized approach to the description of distributed systems, which allows us to achieve abstraction providing the possibility of adaptation between systems during execution, based on the semantic mechanism of judgments. As a result of experimental verification, the authors noted that the ontology-based approach reduced the complexity of SoS development by abstracting system heterogeneity through comprehensive description of model structure and autonomous manipulation during program execution.

Taking into account the identified problems and shortcomings of existing approaches and methods of adaptation for complex systems, the problem of improving the process of system adaptation of software by modifying graphic components and functional modules remains relevant.

The purpose of this work is to design a formalized process of configuring adaptive software depending on user requirements and features of the software environment for various devices, as well as to substantiate the possibility of increasing the reliability and efficiency of software systems by applying an ontology-based adaptation mechanism.

IV. RESEARCH METHODOLOGY

We propose the use of an ontological approach as the basis for the implementation of the mechanism of adaptation of software systems.

The primary task is to define the specification of conceptual requirements (Figure 1):

- the presence of an ontology that allows you to store the architecture of software products;
- adaptation to the specifics of the subject area of the industry taking into account the context of applied tasks;
- reliability of storage and speed of data processing in the ontology repository;
- availability of a logical conclusion mechanism;
- the presence of a mechanism for adapting the interface for users;
- availability of data import mechanisms from external information resources.

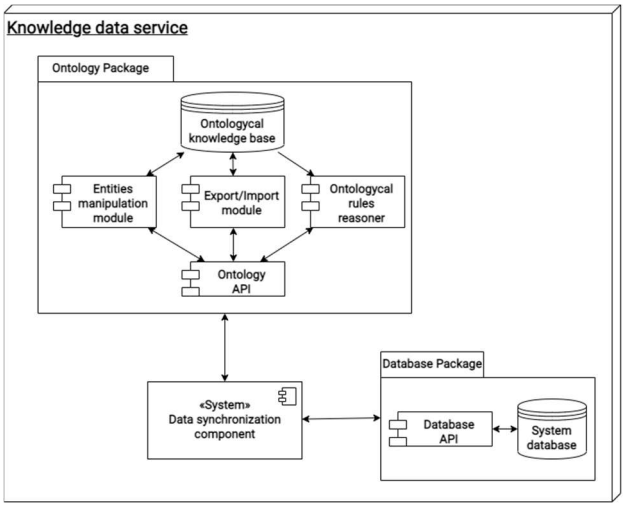

Figure 1. Component structure of the ontological data and knowledge server

In a formalized form, the ontology of an adaptive software system can be represented as a combination of three elements [23]:

$$O_{meta} = \langle C_{meta} \quad R_{meta} \quad Prop_{meta} \rangle, \quad (1)$$

where C_{meta} – set of entities of the adaptive software system subject area containing information about users and possible configurations of the software.

R_{meta} – a set of relations between entities of the subject area.

$Prop_{meta}$ – a set of properties characterizing the essence (concept) of the subject area.

The formed ontology of adaptive software systems contains only the main abstractions, concepts and connections necessary for defining the software configuration. Such a structure allows you to abstract from the specific implementation of software for different subject areas. In addition, the model developed provides the possibility of structural expansion of basic concepts by using the process of imitation ("has-a" relationship), as well as supplementing the model with new relationships and rules. In a detailed form, the conceptual model of the adaptive software system is presented in Figure 2.

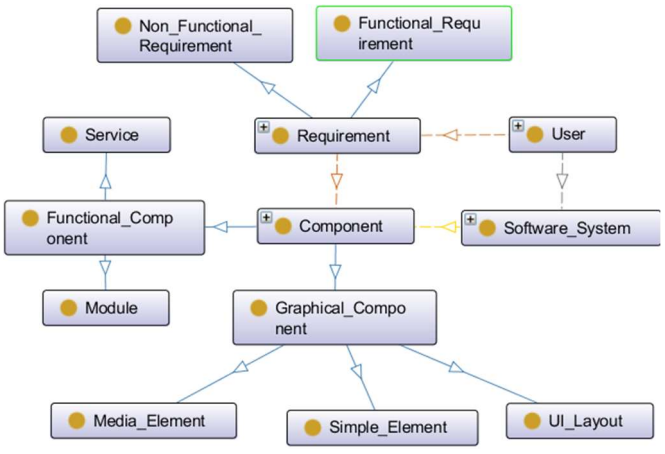

Figure 2. Conceptual ontological model of an adaptive software system

The ontological model of the subject area for self-adaptive systems allows determining the necessary actions to respond to changes in the requirements or needs of the user. However, this model is only part of the process of generating software settings.

The functional model of the proposed process of dynamic software adaptation is presented in Fig. 3.

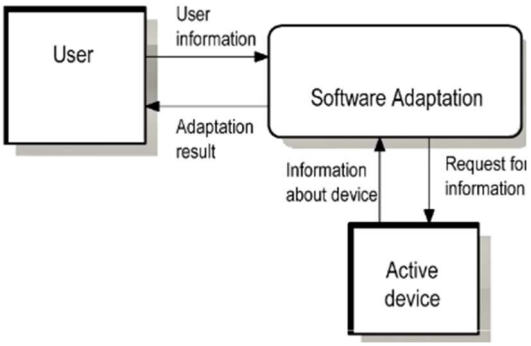

Figure 3. Conceptual diagram of the modification of the self-adaptive software system

The interaction of the general process of software adaptation with external entities is presented at the top level of the hierarchy. The system receives information about the user and provides the user with the modification results depending on the specification of the active device.

The main external objects for the process are "User" and "Active Device". Actually, to initiate the process of changing the settings, the user sends selected information, which serves as an identifier for the necessary changes in the operation or appearance of the system. Adaptive structure of the ontological model as well as adaptive software design allows one to dynamically define settings and selected information. For example, mobile application for people with speech and hearing impairments could provide the form or list of questions that would determine which problems and requirements user has. Based on this selected information future adaptation will proceed. During processing, the system makes a return request for information about the active device, which will help identify the possibility of changes in specific characteristics. The generated settings are taken into account during the process of modifying the system components, and the adapted characteristics are presented to the user.

In the case of complex systems, a hierarchy of context diagrams is built. Such a hierarchy should be described using IDEF0 and DFD models, which allow describing the business processes occurring at each level of the hierarchy and the data flows between these processes. In our case, since the "Software adaptation" process is complex and can be represented as a sequence of sub-processes, it is advisable to carry out its decomposition.

The detailed process of modification of self-adaptive systems, which is a decomposition of the "Software adaptation" process, is presented in Fig. 4.

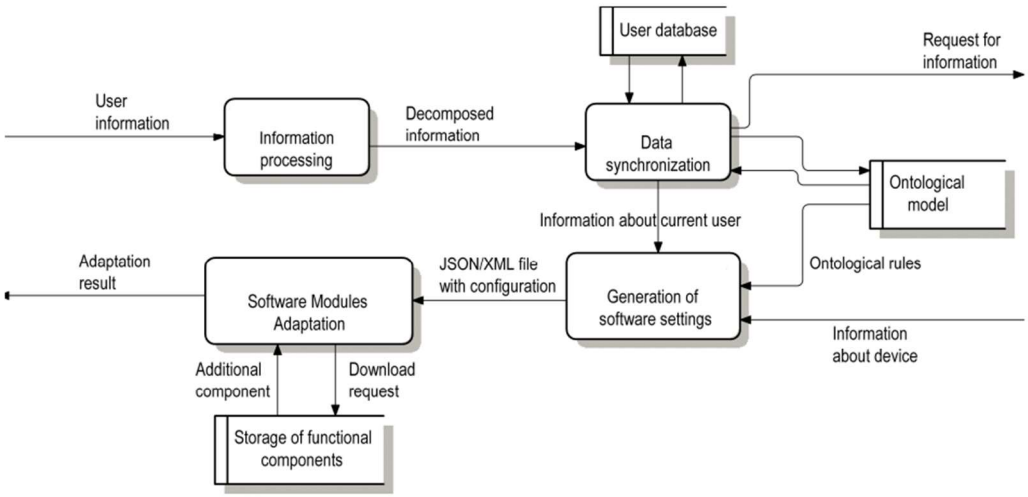

Figure 4. Data flow diagram of the "Software adaptation" process

According to this modification process, the method of changing system components consists of the following stages:

1. Processing of information received from the user (user identification, checking the correctness of filling in information).
2. Data synchronization, which allows you to create or modify the necessary records both in the database and in the ontological knowledge base.
3. Settings generation – the stage processes data about the selected user and forms the necessary system settings, based on the relationships and properties of the concepts of the ontology.
4. Adaptation of the software system includes modification of functional characteristics and graphical interface based on a set of established settings.

In accordance with the presented decomposition, the information received from the user is processed and synchronized with the knowledge base. After synchronization, the settings are generated, which allows you to adapt the software product.

The "Software Adaptation" process shows the general communication between subprocess and data storage from the beginning of the request to generation of the response. A detailed description of the sequence of data transmission

between system elements for dynamic adaptation of the existing software solution is revealed in the process "Software Modules Adaptation", the decomposition of which is presented in Figure 5.

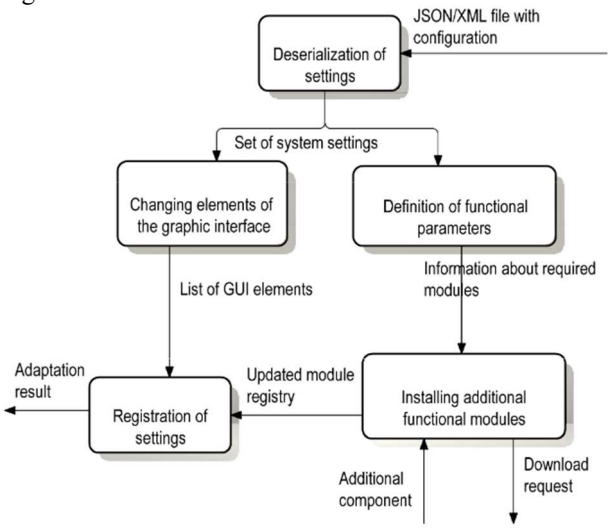

Figure 5. Decomposition of the "Software Modules Adaptation" process

The process of software product adaptation consists of several stages:

1. "Deserialization of settings", this stage consists in extracting data from the JSON document for their further registration in the permanent storage of the software.
2. "Changing elements of the graphic interface", that is, applying parameters to the graphic elements of the software, installing new styles and loading additional resources (language packs, images, fonts, etc.).
3. "Definition of functional parameters" is based on the obtained parameters, the methods of loading and applying the relevant modules, services or routines are determined.
4. "Installing additional functional modules", this stage is aimed at installing and registering additional modules or services in the system.
5. "Registration of settings" is the final stage of adaptation, which ensures correct saving and display of new software settings.

Thus, according to the proposed adaptation process, the ontology model is used in two key stages: data synchronization and generation of software settings. The first stage allows you to store the selected information about the user in the ontological knowledge base in order to facilitate further requests regarding the modification of the system characteristics. The second stage actually starts the semantic mechanism of judgments, which begins processing the semantic rules defined in the ontological model. These two stages are interconnected, since the result of data synchronization between the database and the knowledge base is responsible for the correct filling of the knowledge base only with updated information.

V. CASE STUDY

The adaptation process according to our proposed approach takes place after checking the user data and registering them in the ontological knowledge base. To create personalized system settings, instances of the relevant concepts of the ontological model are created, and connections between the created concepts of the subject area are established. On the basis of semantic rules, decisions are made about the need to assign specific adaptation parameters. The result of this approach is a set of parameters that are used to personalize the system [23]. When modifying a mobile application, two separate processes take place in parallel: personalization of the graphical interface and modification of the system functionality.

According to the classical approach, which was presented in our previous studies, the formation of the model and settings of the software system consists in determining the specific entities of the subject area. Despite the high processing speed, this approach loses efficiency if the structure of the ontological model is variable. As can be seen from our previous research [24], the classical approach to determining optimal system settings depends linearly on the total number of entities of the ontological model. However, performance is lost if it is necessary to expand the software system with additional modules, because it is necessary to change the structure of the ontological model [26, 27], which in turn creates additional connections between elements.

The approach proposed in this work eliminates this problem. The improved ontological model defines the main abstractions among the concepts and entities of the subject area. Such a solution, in turn, allows the structure of the ontological model to remain unchanged, and its expansion is ensured by the creation of new instances of existing entities [25].

In order to compare the proposed approach and the classic version, the software system developed in the previous study was used [24]. The system was designed according to the principles of a three-tier and plug-in architecture, which consists of an Android application and a web server implemented using the Flask framework. The plug-in based software design allowed us to change the ontology and method of modification with new implementation while it is still possible to use adaptation rules of both approaches.

To compare the speed of the process of determining the optimal characteristics for the classic implementation and our proposed approach of abstract description of entities, the duration of processing ontological rules is determined. To determine the duration of adaptation, 2 mobile devices and 3 emulators are used, which makes it possible to test the system on various versions of Android. The duration of the generation of settings is determined from the moment of sending the request until the final formation of the parameters. Testing of the system is performed on each device 5 times, in order to determine the worst-case adaptation time depending on the number of entities of the ontological model. The results of adaptation are presented in Table 1.

Table 1. The results of comparing the duration of determining the configuration for system parameters

№, number of instances		100	150	250	450	850	1650	3300
Classic approach	time, t1 (sec)	2.55	3.08	3.46	3.97	5.16	6.78	11.5
	time, t2 (sec)	2.65	2.84	3.14	4.17	4.69	6.63	11.4
	time, t3 (sec)	2.61	2.92	3.12	3.68	4.46	6.67	11.8
	time, t4 (sec)	2.74	2.84	3.13	3.58	4.58	6.65	11.2
	time, t5 (sec)	2.52	2.86	3.17	3.91	4.85	6.02	11.8
	Average time, t avg (sec)	2.61	2.91	3.20	3.86	4.75	6.55	11.54
Abstract approach	time, t1 (sec)	2.30	2.73	2.66	3.24	3.23	3.99	6.55
	time, t2 (sec)	2.20	2.21	2.82	3.05	3.47	4.39	6.96
	time, t3 (sec)	2.52	2.23	2.61	2.80	3.64	4.40	6.66
	time, t4 (sec)	2.55	2.85	2.75	2.94	3.59	4.32	6.59
	time, t5 (sec)	2.65	2.41	2.68	2.96	3.34	4.46	7.47
	Average time, t avg (sec)	2.44	2.49	2.70	3.00	3.45	4.31	6.85

A graphical display of the comparison results is presented in Figure 6. From the presented histogram, it can be seen that the new approach in all tests showed better performance results compared to the classical approaches. The obtained results also demonstrate an improvement in the speed of determining the optimal characteristics of the software, on average, by 20%. In addition, the new method shows better results with an increase in the number of elements and concepts of the ontological model.

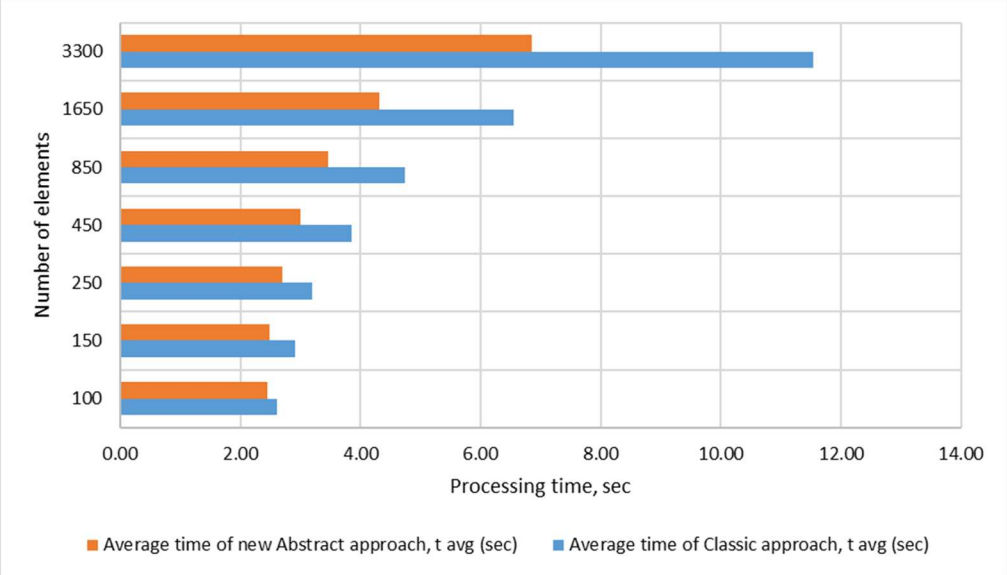

Figure 6. Comparison of the duration of determining the configuration of system parameters

VI. DISCUSSION

The proposed adaptation process provides the possibility of dynamic modification of the software structure and the formation of a set of functional and graphic components of the system based on the user's preferences and requirements, as well as using information about currently active device. At the same time, it should be noted that the presented abstract approach to the description of the subject area and software configuration also solves the problem of adding and registering new functional components in the system. This feature is ensured by the implementation of their abstract representation and the use of designed interfaces, which allow connecting modules with a lower level of dependence and connectivity.

The use of the ontological model, which represents the main elements of the software together with the requirements and needs of users, in combination with the proposed method allows improving the process of modification of software components. This combination makes it possible to distinguish the adaptation process in two stages:

- The first stage is responsible for the synchronization of information between the ontological knowledge base and the database. After the synchronization process, a semantic decision-making engine is launched, which allows determining the optimal settings and characteristics of the system for the user based on ontological rules and user requirements.

- The second stage is software adaptation and modification of system components in accordance with the received settings. This process takes place already on the user's active device, information about which, together with information about the software environment, is also used when modifying components.

Analyzing the proposed approach of software adaptation using the ontological model, it should be noted that in further research it is necessary to improve the first stage of adaptation, namely, the processing of information by the ontological knowledge base and its synchronization with the database. Currently, this stage takes place in an external environment from the user's active device, which in turn requires a constant connection to the network. The solution to such a problem can be the formation of a local package with data about the ontological model and its semantic rules. However, in this case, a problem arises in maintaining and synchronizing the ontological knowledge base.

VII. CONCLUSIONS

An analysis of research aimed at the design and development of adaptive and self-adaptive software systems has been carried out. Existing approaches focus on software adaptation using methods based on feedback diagrams, methods based on machine learning, and combinations thereof. At the same time, the main problem with such adaptation remains the modification of the software structure by adding new functional or graphic components. Another problem of these approaches is taking into account the possibility of modifying software components on different hardware and software platforms.

A formalized approach to modifying the configuration of adaptive software is proposed, taking into account the preferences and requirements of a specific user, as well as the features of currently active device. The designed process uses an ontological model as a knowledge base about the requirements and structure of the software and allows determining the necessary system settings based on the semantic decision-making mechanism. In addition, the configuration change process allows modification of existing components that are registered in the software environment, as well as adding new functional and graphical elements to expand the capabilities of the software.

Using the proposed approach, an experimental study of the speed of the process of determining optimal system characteristics based on ontological models is conducted. The obtained results demonstrate 20% better indicators of the speed of generating software settings, using an ontological model based on an abstract description of concepts, compared to classical approaches to the representation of the subject area. It should also be noted that for 3300 entities and more, it is better to use the proposed approach, since the processing time is reduced from 11.5 to 6.85 seconds on average.

Analyzing the obtained results, it should be noted that in further research it is necessary to improve the processing of information by the ontological knowledge base and its synchronization with the database. This will reduce the number of requests to the ontological model and save the already generated configurations.

References

[1] C. Szabo, B. Sims, T. Mcatee, R. Lodge and R. Hunjet, "Self-adaptive software systems in contested and resource-constrained environments: Overview and challenges," *IEEE Access*, vol. 9, pp. 10711-10728, 2021, https://doi.org/10.1109/ACCESS.2020.3043440.

[2] R. D. Lemos et al., "Software engineering for self-adaptive systems: A second research roadmap," *Software Engineering for Self-Adaptive Systems II*, Berlin, Germany: Springer, 2013, pp. 1–32.

[3] F. Macías-Escrivá, R. Haber, R. del Toro and V. Hernandez, "Self-adaptive systems: A survey of current approaches, research challenges and applications", *Expert Systems with Applications*, vol. 40, no. 18, pp. 7267-7279, 2013. https://doi.org/10.1016/j.eswa.2013.07.033.

[4] I. Gerostathopoulos, T. Vogel, D. Weyns and P. Lago, "How do we evaluate self-adaptive software systems? A ten-year perspective of SEAMS," *Proceedings of the 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, Madrid, Spain, 2021, pp. 59-70, https://doi.org/10.1109/SEAMS51251.2021.00018.

[5] D. Weyns, *Software Engineering of Self-adaptive Systems*, In Handbook of Software Engineering, Springer, 2019, pp. 399–443. https://doi.org/10.1007/978-3-030-00262-6_11.

[6] C. Krupitzer, T. Temizer, T. Prantl, and C. Raibulet, "An overview of design patterns for self-adaptive systems in the context of the internet of things," *IEEE Access*, vol. 8, pp. 187384–187399, 2020. https://doi.org/10.1109/ACCESS.2020.3031189.

[7] T. R. D. Saputri and S.-W. Lee, "The application of machine learning in self-adaptive systems: A systematic literature review," *IEEE Access*, vol. 8, pp. 205948-205967, 2020, https://doi.org/10.1109/ACCESS.2020.3036037.

[8] O. Gheibi, D. Weyns, and F. Quin, "Applying machine learning in self-adaptive systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 15, no. 3, pp. 1–37, 2020. https://doi.org/10.1145/3469440.

[9] D. Papamartzivanos, F. Gomez Marmol, and G. Kambourakis, "Introducing deep learning self-adaptive misuse network intrusion detection systems," *IEEE Access*, vol. 7, pp. 13546–13560, 2019. https://doi.org/10.1109/ACCESS.2019.2893871.

[10] L. Wang, "Search-based adaptation planning framework for self-adaptive systems," *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Buenos Aires, Argentina, 2017, pp. 465-466, https://doi.org/10.1109/ICSE-C.2017.21.

[11] J. Wan, Q. Li, L. Wang, L. He and Y. Li, "A self-adaptation framework for dealing with the complexities of software changes," *Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, 2017, pp. 521-524, https://doi.org/10.1109/ICSESS.2017.8342969.

[12] Y. Li, Q. Li, L. Wang, W. Cheng and T. Wu, "ADAPT: An agent-based development toolkit and operation platform for self-adaptive systems," *Proceedings of the 2017 IEEE Conference on Open Systems (ICOS)*, Miri, Malaysia, 2017, pp. 53-58, https://doi.org/10.1109/ICOS.2017.8280274.

[13] J. Mertz and I. Nunes, "On the practical feasibility of software monitoring: A framework for low-impact execution tracing," *Proceedings of the 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, Montreal, QC, Canada, 2019, pp. 169-180, https://doi.org/10.1109/SEAMS.2019.00030.

[14] E. Zavala, "Towards adaptive monitoring services for self-adaptive software systems," *Proceedings of the Service-Oriented Computing – ICSOC 2017 Workshops*, 2018, pp. 357–362. https://doi.org/10.1007/978-3-319-91764-1_31.

[15] L. Schölkopf, M.-M. Wolf, V. Hutmann, and F. Diermeyer, "Conception, development and first evaluation of a context-adaptive user interface for commercial vehicles," *Proceedings of the 13th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 2021. https://doi.org/10.1145/3473682.3480256.

[16] D. Nehurytsia and E. Sokolova, "User model in adaptive web-oriented system software interfaces," *Radio Electronic and Computer Systems*, vol. 1, no. 65, pp.104-111, 2014.

[17] I. Tvoroshenko, M. A. Ahmad, S. K. Mustafa, V. Lyashenko, A. R. Alharbi, "Modification of models intensive development ontologies by fuzzy logic," *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 3, pp. 939–944, 2020. https://doi.org/10.30534/ijeter/2020/50832020.

[18] A. Kindo, G. Kaladzavi, S. Malo, G. Camara, T. M. Y. Tapsoba and Kolyang, "Fuzzy logic approach for knowledge modeling in an ontology: A review," *Proceedings of the 2020 IEEE 2nd International Conference on Smart Cities and Communities (SCCIC)*, Ouagadougou, Burkina Faso, 2020, pp. 1-8, https://doi.org/10.1109/SCCIC51516.2020.9377335.

[19] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of Systems Engineering," *ACM Computing Surveys*, vol. 48, no. 2, pp. 1–41, 2015. https://doi.org/10.1145/2794381.

[20] D. S. Santos, B. R. Oliveira, R. Kazman, and E. Y. Nakagawa, "Evaluation of systems-of-systems software architectures: State of the art and future perspectives," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–35, 2022. https://doi.org/10.1145/3519020.

[21] J. Axelsson, J. Fröberg, and P. Eriksson, "Architecting systems-of-systems and their constituents: A case study applying industry 4.0 in the construction domain," *Systems Engineering*, vol. 22, no. 6, pp. 455–470, 2019. https://doi.org/10.1002/sys.21516.

[22] A. Elhabbash, V. Nundloll, Y. Elkhatib, G. S. Blair, and V. S. Marco, "An ontological architecture for principled and automated system of systems composition," *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*,. 2020. https://doi.org/10.1145/3387939.3391602

[23] D. Fedasyuk and I. Lutsyk, "Method of modification of self-adaptive software systems based on ontology," *Proceedings of the 2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, Lviv-Slavske, Ukraine, 2022, pp. 530-533, https://doi.org/10.1109/TCSET55632.2022.9766856.

[24] D. Fedasyuk and I. Lutsyk, "Tools for adaptation of a mobile application to the needs of users with cognitive impairments," *Proceedings of the 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2021, pp. 321-324, https://doi.org/10.1109/CSIT52700.2021.9648702.

[25] D. Fedasyuk and I. Lutsyk, "The use of ontology in the process of designing adaptive software systems," *Proceedings of the 2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT)*, Lviv, Ukraine, 2022, pp. 503-506, https://doi.org/10.1109/CSIT56902.2022.10000528.

[26] M. Moshref, R. Al-Sayyad, "Developing ontology approach using software tool to improve data visualization (Case study: Computer Network)," International Journal of Modern Education and Computer Science (IJMECS), vol. 11, no. 4, pp. 32-39, 2019. https://doi.org/10.5815/ijmecs.2019.04.04.

[27] H. Razouki, "Security policy modelling in the mobile agent system," International Journal of Computer Network and Information Security (IJCNIS), vol. 11, no. 10, pp. 26-36, 2019. https://doi.org/10.5815/ijcnis.2019.10.04.

***Dmytro FEDASYUK,** Professor, Head of Software Engineering Department, Institute of Computer Sciences and Information Technologies, Lviv Polytechnic National University.*

***Research interests:** mathematical modeling and information technologies, modeling of thermal regimes in microelectronic systems, software design*

***Illia LUTSYK,** a PhD student of Software Engineering Department, Institute of Computer Sciences and Information Technologies, Lviv Polytechnic National University.*

***Research interests:** adaptive software, ontological models, software design*