http://doi.org/10.35784/iapgos.6656 *received: 03.10.2024 | revised: 20.11.2024 | accepted: 13.12.2024 | available online: 21.12.2024*

# SYNCHRONIZATION OF EVENT-DRIVEN MANAGEMENT DURING DATA COLLECTION

**Valeriy Kuzminykh[1], Oleksandr Koval[1], Yevhen Havrylko[1], Beibei Xu[1], Iryna Yepifanova[2], Shiwei Zhu[1], Nataliia Bieliaieva[3], Bakhyt Yeraliyeva[4]**

[1]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Department of Software Engineering in Energy, Kyiv, Ukraine, [2]Vinnytsia National Technical University, Vinnytsia, Ukraine, [3]Dragomanov Ukrainian State University, Kyiv, Ukraine, [4]M. Kh. Dulaty Taraz Regional University, Taraz, Kazakhstan

***Abstract.*** *The article considers an approach to implementing the architecture of a microservice system for processing large volumes of data based on the event-oriented approach to managing the sequence of using individual microservices. This becomes especially important when processing large volumes of data from information sources with different performance levels when the task is to minimize the total time for processing data streams. In this case, as a rule, the task is to minimize the number of requests for information sources to obtain a sufficient amount of data relevant to the request. The efficiency of the entire software system as a whole depends on how the microservices that provide extraction and primary processing of the received data are managed. To obtain the required amount of relevant data from diverse information sources, the software system must adapt to the request during its operation so that the maximum number of requests are directed to sources that have the maximum probability of finding the data necessary for the request in them. An approach is proposed that allows adaptively managing the choice of microservices during data collection and by emerging events and, thus, forming a choice of information sources based on an assessment of the efficiency of obtaining relevant information from these sources. Events are generated as a result of data extraction and primary processing from certain sources in terms of assessing the availability of data relevant to the request in each of the sources considered within the framework of the selected search scenario. Event-oriented microservice architecture adapts the system operation to the current loads on individual microservices and the overall performance by analyse the relevant events. The use of an adaptive event-oriented microservice architecture can be especially effective in the development of various information and analytical systems constructed by real-time data collection and design scenarios of analytical activity. The article considers the features of synchronous and asynchronous options in the implementation of event-oriented architecture, which can be used in various software systems depending on their purpose. An analysis of the features of synchronous and asynchronous options in the implementation of event-oriented architecture, their quantitative parameters, and features of their use depending on the type of tasks is carried out.*

**Keywords**: Big Data, microservices, adaptation, event-driven software architecture, information technology, ontology

## SYNCHRONIZACJA ZARZĄDZANIA STEROWANEGO ZDARZENIAMI PODCZAS GROMADZENIA DANYCH

***Streszczenie.*** *W artykule rozważono podejście do implementacji architektury systemu mikrousług do przetwarzania dużych ilości danych w oparciu o podejście zorientowane na zdarzenia do zarządzania sekwencją korzystania z poszczególnych mikrousług. Staje się to szczególnie ważne podczas przetwarzania dużych ilości danych ze źródeł informacji o różnych poziomach wydajności, gdy zadaniem jest zminimalizowanie całkowitego czasu przetwarzania strumieni danych. W tym przypadku, co do zasady, zadaniem jest zminimalizowanie liczby żądań do źródeł informacji w celu uzyskania wystarczającej ilości danych istotnych dla żądania. Wydajność całego systemu oprogramowania jako całości zależy od sposobu zarządzania mikrousługami, które zapewniają ekstrakcję i podstawowe przetwarzanie otrzymanych danych. Aby uzyskać wymaganą ilość odpowiednich danych z różnych źródeł informacji, system oprogramowania musi dostosować się do żądania podczas jego działania, tak aby maksymalna liczba żądań była kierowana do źródeł, które mają maksymalne prawdopodobieństwo znalezienia w nich danych niezbędnych do żądania. Zaproponowano podejście, które pozwala adaptacyjnie zarządzać wyborem mikrousług podczas gromadzenia danych i pojawiających się zdarzeń, a tym samym kształtować wybór źródeł informacji w oparciu o ocenę skuteczności uzyskiwania odpowiednich informacji z tych źródeł. Zdarzenia są generowane w wyniku ekstrakcji danych i przetwarzania pierwotnego z określonych źródeł w zakresie oceny dostępności danych istotnych dla żądania w każdym ze źródeł uwzględnionych w ramach wybranego scenariusza wyszukiwania. Architektura mikrousług zorientowana na zdarzenia dostosowuje działanie systemu do bieżących obciążeń poszczególnych mikrousług i ogólnej wydajności poprzez analizę odpowiednich zdarzeń. Wykorzystanie adaptacyjnej architektury mikrousług zorientowanej na zdarzenia może być szczególnie skuteczne w rozwoju różnych systemów informacyjnych i analitycznych zbudowanych w oparciu o gromadzenie danych w czasie rzeczywistym i projektowanie scenariuszy działalności analitycznej. W artykule rozważono cechy opcji synchronicznych i asynchronicznych w implementacji architektury zorientowanej na zdarzenia, które mogą być wykorzystywane w różnych systemach oprogramowania w zależności od ich przeznaczenia. Przeprowadzono analizę cech opcji synchronicznych i asynchronicznych w implementacji architektury zorientowanej na zdarzenia, ich parametrów ilościowych oraz cech ich wykorzystania w zależności od rodzaju zadań.*

**Słowa kluczowe**: Big Data, mikrousługi, adaptacja, architektura oparta na zdarzeniach, technologia informacyjna, ontologia

## Introduction

The collection of information based on certain characteristics aimed at fulfilling a request from open information sources of different composition and status has become one of the most common methods for obtaining information in various spheres of activity, including scientific, commercial, social, public, and state. As a rule, for this, experts collect and analyze information from mass media, public reports, official data, materials of press conferences, public statements, professional and academic reports, conferences, reports, and articles, while using all available sources regardless of the degree of completeness and relevance of data. The use of electronic information carriers largely determines and stimulates the development of approaches and methods of directed search and increases the efficiency of both individual search procedures and methods of organizing software tools focused on collecting and processing the necessary information.

A feature of collecting information from open sources is the uncertainty and instability of the information content of these sources, the lack of substantiated and more or less reliable a priori information about their content, its relevance, and a large volume of data [7]. In advance, the low accuracy and effectiveness of expert assessments of the correspondence of data from these sources to the topics of inquiries is characteristic, which does not allow for a more or less reasonable choice of certain sources for extracting and obtaining the necessary information upon request [11]. Therefore, to extract the required data from open sources of information, it is necessary to select the essential information from various possible and available sources and further effectively consolidate data from many sources using specialized software tools that should ensure:

- automatic selection of the most relevant, according to the request, sources of information during the collection of information;
- the possibility of accumulating and analyzing information about the state of the sources during the execution of the request for further correction of information collection procedures;
- formation of typical scenarios for the collection and primary processing of information by request and taking into account

the possibility of changing the state of the sources upon repeated request [16, 25];
- analysis of both the most promising sources from the point of view of relevance, completeness, and relevance, and less relevant sources;
- construction of a certain assessment of promising sources from the point of view of relevance for their arrangement.

The solution of such a set of problems requires the development of software systems complex in terms of architecture, that can effectively solve the tasks of searching and analyzing such large amounts of data [17, 27]. Such software systems must perform a large volume of data extraction from a significant number of sources, their initial processing, and the determination of the relevance of the received data with an assessment of the prospective use of sources [7, 24].

The use of a monolithic architecture of a software system to solve problems of this type cannot be efficient enough, both from the point of view of developing and testing such a system and from the point of view of implementing parallel processes of collection and processing. They can be cumbersome to work with when you need to add new features, make changes, or even remove some unnecessary features.

In addition, the following important features and properties of monolithic architecture can be noted:
- A monolithic application is faster and easier to implement when it comes to creating a single application that contains almost all the necessary elements from the ground up.
- Through close interaction, the entire application is deployed or updated on the server or in the cloud for any change.
- The entire application must be scaled, even if only one capability or characteristic needs to be scaled.
- You have to redeploy the entire program if something needs to be changed.
- It is difficult to test the system because it is necessary to test the entire software system at once.
- It is difficult to make changes to the program because changes in one place can have undefined consequences in other parts.

Microservices architecture is quite a good option for cases where complex multi-functional large-scale applications are developed and periodic updating of processing methods is required, when different teams work independently on different parts of the software and when there are different domains to be processed and to combine separate results [8, 15].

Microservice architecture has quite a lot of properties that distinguish the results of its use from monolithic [13, 23]:
- Developments are slower than monolithic ones because each service is developed as a separate software module.
- Each service has its data store and, is responsible for a specific domain, and can be developed, modified, or deployed independently.
- Services have a smaller size, which makes them easily adaptable to new changes, improvements, and fixes.
- It is easy to scale individual services without compromising others.
- There is no need to stop the entire program if you need to make changes to a separate service.
- Individual services are much easier to test.
- Individual services can be quite simply updated and replaced according to current tasks.

All this makes the development of a primary information collection and processing system based on the use of microservice architecture more efficient, rational, and attractive for solving the problems of collecting and processing information from disparate sources.

The use of microservice architecture for the construction of information collection and primary processing systems provides significant additional opportunities and significantly expands the conditions of their use, regardless of the complexity of the request and the number of necessary sources used to extract information relevant to the request.

When building information-analytical systems and information collection and primary processing systems based on the use of microservices, it is possible to use query execution scenarios that can adapt to current results and the state of the process of interaction with data sources.

Management of microservices can be quite complicated in connection with a significant number of microservices in the system, different in terms of functions and implementation features, and the need to define clear interactions and the sequence of their use. Each microservice must have the necessary computing capabilities, data storage and some additional specific resources for microservices. All microservices in a software system must be suitable for the ability to control them during the use of the software system 2

The most common ways of managing microservices today are containerization and virtualization, which can be implemented by quite a variety of methods.

Docker's containerization management and deployment automation system have contributed significantly to the fact that containers have become particularly popular [9]. This approach in the interaction of microservices allows you to isolate applications from each other, thus providing them with significant independence in performing their functions and the possibility of parallelizing computing processes. Containers typically use the server's current operating system. This makes it possible to ensure the distribution of resources between containers. Using containers provides a low implementation cost and short start-up time with low resource requirements.

Containers greatly simplify the work of both programmers and program development managers. Containers make it possible to pack both the program and all its necessary components into a single image. These can be both library system utilities and necessary configuration files. This greatly simplifies the deployment and migration of the application.

Containers make application deployment easier. In the usual version, to install some programs, you need to perform standard actions, such as executing a scenario, configuring files, and other necessary elements. Containers make it possible to completely automate this process, as they include the entire sequence and the composition of the execution of actions.

Containers also simplify, if necessary, deployment on multiple servers. Normally, to deploy the same application to multiple machines on the network, you would need to repeat the same steps. Containers eliminate this unnecessary routine work and allow you to automate the entire deployment.

Sharing an operating system with containers can lead to some complications and significant risks (danger). This can be very important in specialized information collection and processing systems that have certain restrictions on access to information. The use of containers requires that all applications included in the system can run in a virtual machine of the operating system of the server on which they are deployed.

One of the main concerns with this is security, as containers share access to the host machine's operating system. If the server uses containers with only one company's services, this is unlikely to create a problem, but modern approaches to the joint rental of cloud resources require paying more attention to the placement of containers when implementing a software system.

Deploying microservices on virtual machines does not have most of the problems and disadvantages of containers, although their use is more complex and requires much more time to implement. This can be quite critical for solving the problems of gathering and primary processing of information. As a rule, virtual machines provide complete isolation with a standalone operating system at the expense of hardware specific to each instance.

Although hosting microservices on virtual machines provides higher security than using containers, it requires much more implementation overhead. Each virtual machine to be used to host microservices has a higher cost compared to containers, a significantly longer start up time, and higher system resource requirements for operation.

To date, there are already many developments that allow increasing the efficiency of the use of virtual machines, reducing the time and cost of implementing the microservice architecture of software applications on their platform.

In cases where the need to implement security requirements is in the first place, then placing microservices on virtual machines can have significant advantages when choosing a way to implement a software system. Therefore, according to the development and improvement of these technologies, virtual machines can become a more competitive alternative to containers in supporting microservices architecture.

As a rule, the tasks of managing containers and virtual machines are solved with the help of specialized programs of different structures and complexity, which are constantly being developed by different companies, both for their needs and for the needs of the market [12, 14].

Container management software systems manage both container deployment and resource allocation and integration with computing resources. Microservices must be able to scale up and down based on workloads and performance requirements.

Container management software systems must support vertical scaling, in which the load on computing and network resources such as CPU, memory disk, network, and external communication channels can increase or decrease on each microservice instance. Horizontal scaling should also be supported with the addition or removal of new instances depending on the need to adapt the management scenario of the microservice system according to the request it performs [5, 6].

One of the most relevant and functional types of software architecture that can be used to solve the problems of gathering information from open information sources of different composition and status is event-driven architecture (Event-driven architecture, EDA) [10, 22].

Event-driven architecture from the point of view of implementation can be built both on the use of virtual machines and on containerization, depending on the specific capabilities and wishes of customers and developers. To solve the problems of collecting information according to certain characteristics from information sources of different composition and status, the software architecture built based on container management can be more effective and appropriate to the specifics of the tasks.

Event-driven architecture is one of the types of software architecture that takes into account the events that occur and are analyze during the operation of the software system, and the reaction of the software system's operation scenario to them. At the same time, the event is interpreted as some action that initiates a message or a necessary change in the application. At the same time, the event can cause significant changes in the state of the software system.

In practice, event-driven architecture is also often seen as a logical development of an adaptive superstructure over microservice architecture. In an event-driven architecture, the focus shifts to events and how they affect the scenario execution processes in the system. The logic of the event-oriented architecture is built according to two types of topologies – Broker and Mediator, named after the intermediary programs that combine the generator and consumer of events.

The main advantages of event-driven architecture are the ability to obtain results in real time; shorter delays in data storage and transmission; greater bandwidth; simple scalability; and high resistance to failures [3, 28].

Modern event-driven micro-service architectural approaches to the management of microservices according to the performance indicator still do not have universal solutions, and the development of this direction has a significant perspective and interest.

Architectural solutions, which are based on event-driven micro-service software architecture, provide an opportunity to significantly expand the analytical, managerial, and control capabilities of information and analytical systems. This, in turn, provides a qualitatively new level of construction, maximally adapted to the structure and processes of search, analysis, and formation of results in the information environment.

An actual problem remains the task of developing ideologies and approaches for the most effective use of micro-service architecture in solving complex information and analytical problems, which reflects the constant increase in the amount of information that must be processed to solve user requests.

The architecture of a microservice software system, driven by events generated during the operation of the software system, is a popular approach to creating a distributed architecture used to create complex scalable applications [2]. This approach is flexibly adaptable to changes in the conditions of use of the software system and can be used both for small programs and for large, complex software systems. The event-driven architecture consists of significantly separated, purpose-built event-processing components that occur during the operation of the software system and that receive information about emerging events and process this information in accordance with current and previous events [26].

## 1. Problem statement

A certain group of open sources of information is considered, which presumably have the necessary data and can be used for extraction, initial processing, and further consolidation of the necessary data corresponding to the request [21]. Such a group of sources is considered for each specific task and is the basis for determining the search scenario by the user's request for the data retrieval system.

$$D_S = \{D_1, D_2, D_3, \ldots, D_n\}$$

It is assumed that each of the sources has its own distinctive features and technical characteristics regarding the form and structure of their storage, which in turn leads to the need to use separate specific approaches to data extraction. Such a variety of forms and structural characteristics of information storage in sources leads to the urgent need to use separate specific software modules for each of the *n* sources.

Requests for the search of the necessary data from a group of specified information sources contain data characteristics, based on which it is possible to determine compliance with a specific request. For example, to search for bibliographic data, such characteristics as surnames and first names of authors, years of publication, country of author, countries of co-authors, language of publication, keywords, and others can be used. Such characteristics are determined by the request and can be specified, if necessary, by the user of the software system. The number (quantity) and composition are limited only by the capabilities of software applications that retrieve the data of the corresponding request. The request may include the number of relevant data units to be extracted, limitations on the number of requests to information sources, and the total time for obtaining data relevant to the request.

The number (quantity) of queries, query volumes, and the number of repeated query sessions are determined by the results of previous sessions. At the same time, a request session means a certain number of requests to each of the sources, after which the data extraction results are evaluated according to the request characteristics. Very often, the total session size is defined as the total number of requests to all sources in one iteration of the search method implemented in the software system.

The assessment of quality or effectiveness for each of the sources in the session is carried out as an assessment of the relevance of the received data by the characteristics of the request. The specific form of evaluation of the relevant received data depends on the specific characteristics of the request and may be determined by the specifics of the implementation of data extraction algorithms and their primary processing.

The general task is to obtain a sufficient amount of data corresponding to the request in the minimum time or for the minimum number of requests to information sources.

## 2. Description of the algorithm

The development of systems for collecting and processing large flows of data from disparate sources in most cases requires the determination of specific implementation features by the tasks that reflect both the features of the selected sources for data extraction and the features of the requirements for the quality, relevance, and reliability of the information collected from the relevant sources.

The presented structure of the event-oriented micro-service software system (Fig.1) has some features that are related to the use of a separate service, which manages the actions of the system as a whole from the point of view of both the analysis of the performance of tasks following the initial request of the system user and the processes of selecting the most relevant sources of information by activating the corresponding microservices that are mutually unambiguously associated with specific sources [4, 18].

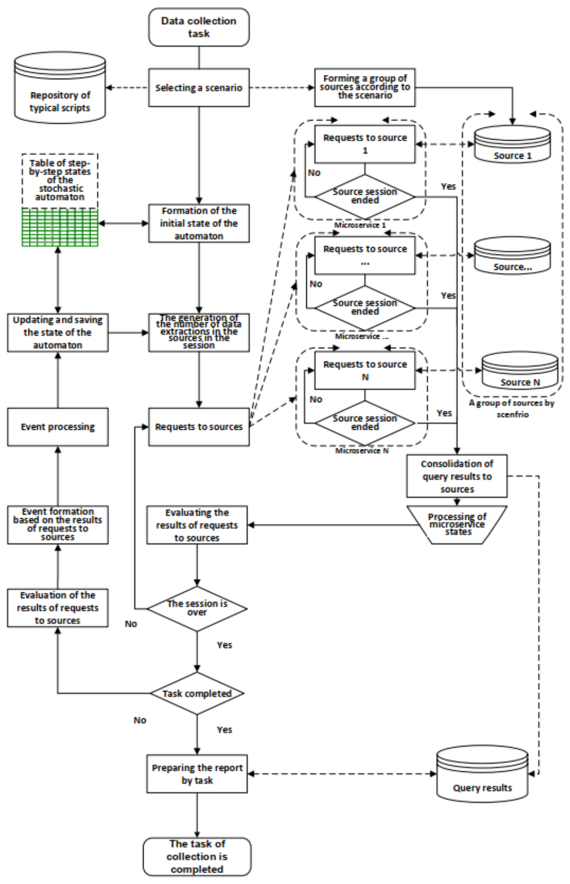

Data collection task
Selecting a scenario
Repository of typical scripts
Forming a group of sources according to the scenario
Table of step-by-step states of the stochastic automaton
Formation of the initial state of the automaton
Requests to source 1
Source session ended
No
Yes
Microservice 1
Source 1
Requests to source ...
Microservice ...
Source...
Requests to source N
Microservice N
Source N
A group of sources by scenfrio
Updating and saving the state of the automaton
The generation of the number of data extractions in the sources in the session
Event processing
Requests to sources
Consolidation of query results to sources
Processing of microservice states
Event formation based on the results of requests to sources
Evaluating the results of requests to sources
Evaluation of the results of requests to sources
The session is over
Task completed
Preparing the report by task
Query results
The task of collection is completed

*Fig. 1. Algorithm of stochastic adaptive event-driven system*

The specificity of the task under consideration requires, as a rule, the development of specific software applications for collecting information from each of the sources considered as potential repositories of the necessary information. At the same time, the list of such sources can be determined by the corresponding typical scenario, which must be matched to the type and other specific features of the request for data collection and processing in the information and analytical system.

To initiate the operation of the system, it is necessary to formulate an appropriate data collection task, which is the basis for choosing the appropriate scenario. One of the standard scenarios can be used as such a scenario or a new scenario defined by an analyst who is a user of this system [19, 20].

The task, with such consideration, for example, within the framework of the information-analytical system of the analysis of scientific activity, may include such parameters of data selection, i.e., assessment of their relevance, as key data, i.e. authors' surnames, topics, keywords, periods time, dates, names of countries, scientific areas, etc. Based on this task, a form of request to data sources is formed, which is the basis for determining the correspondence of data in the source to the task of information search.

In addition, the task may include weighting factors corresponding to each of the parameters of the task. For the operation of the system, a repository of typical scenarios is created, which stores pre-prepared scenarios and such scenarios that were refined and obtained after the use of previous typical scenarios, as a result of the software system's work on its previous use for the current task or other tasks similar in structure. Initial scenarios can be built and entered by the system user and saved for future use.

The process of building typical scenarios requires the creation of a scenario model (a model of data collection scenarios or a functional model), a model of production rules, and an executive model of the software system [19, 24].

These models are built based on a previously built ontology of the subject area of the software system, which specifies the description of the main entities (concepts and relations) of the subject area in the form of classes of objects, instances of classes, their properties and relations between classes and properties, including a description of the information sources necessary for data collection tasks (Fig. 2).

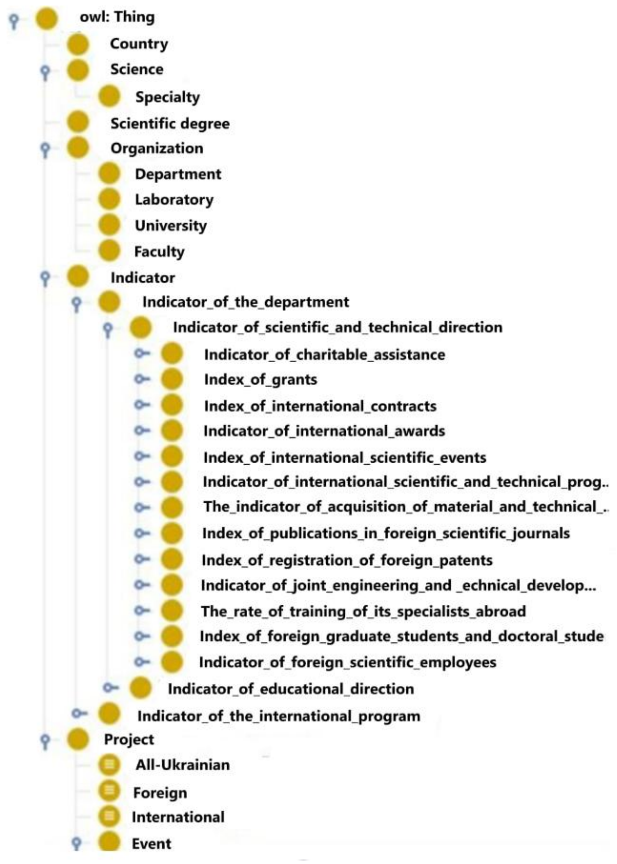

owl: Thing
- Country
- Science
  - Specialty
- Scientific degree
- Organization
  - Department
  - Laboratory
  - University
  - Faculty
- Indicator
  - Indicator_of_the_department
    - Indicator_of_scientific_and_technical_direction
      - Indicator_of_charitable_assistance
      - Index_of_grants
      - Index_of_international_contracts
      - Indicator_of_international_awards
      - Index_of_international_scientific_events
      - Indicator_of_international_scientific_and_technical_prog..
      - The_indicator_of_acquisition_of_material_and_technical_..
      - Index_of_publications_in_foreign_scientific_journals
      - Index_of_registration_of_foreign_patents
      - Indicator_of_joint_engineering_and _echnical_develop...
      - The_rate_of_training_of_its_specialists_abroad
      - Index_of_foreign_graduate_students_and_doctoral_stude
      - Indicator_of_foreign_scientific_employees
    - Indicator_of_educational_direction
  - Indicator_of_the_international_program
- Project
  - All-Ukrainian
  - Foreign
  - International
  - Event

*Fig. 2. A fragment of the class hierarchy of the domain ontology*

Using the Protege-5.5.0 ontology editor, an ontology of the subject area of the search for relevant information on the analysis of the scientific activity of the organization was built (Fig. 3), which was checked for completeness and semantic compatibility [7] and used to build typical scenarios.

The scenario model reflects the order and content of information management at the functional level:

$$Sc^M = \cup_i \{TSc_i, ASc_i, Ex_i, G_i | i = \overline{1,N}\}$$

where $TSc_i$ – a typical scenario of the $i$-th task of data collection; $ASc_i$ – an extended (secondary) scenario for the $i$-th task of data collection; $G_i$ – the goal of the $i$-th task execution scenario data collection, which has such characteristics as criteria, time, and resources.

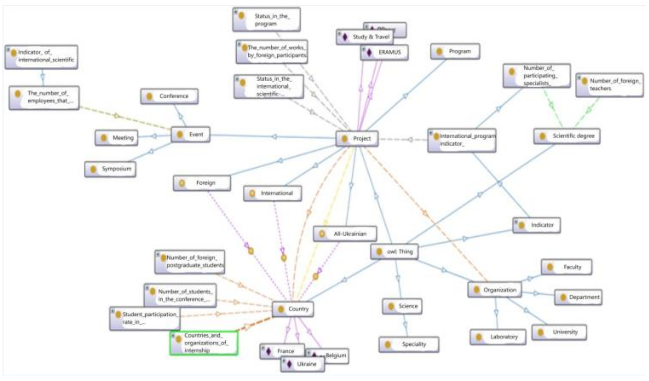

Status_in_the_ program
Study & Travel
ERAMUS
Program
Indicator_of_ international_scientific
The_number_of_works_ by_foreign_participants
Status_in_the_ international_ scientific-...
Number_of_ participating_ specialists_
Number_of_foreign_ teachers
The_number_of_ employees_that_...
Conference
Meeting
Event
Project
International_program indicator_
Scientific degree
Symposium
Foreign
International
Indicator
All-Ukrainian
owl: Thing
Faculty
Number_of_foreign_ postgraduate_students
Organization
Department
Number_of_students_ in_the_conference_...
Country
Science
Student_participation_ rate_in_...
Countries_and_ organizations_of_ internship
Laboratory
University
France
Belgium
Speciality
Ukraine

*Fig. 3. A fragment of the visual representation of the domain ontology in the Protégé 5 editor*

The model of production rules describes the rules of logical inference in terms of classes and relations for performing operations on instances of ontology classes of the subject area of the software system.

The executive model of the software system consists of individual microservices designed for the implementation of typical scenarios and the output of new knowledge, based on which new typical scenarios are built. The scenario model links the algorithms of software implementation of microservices with the model of production rules [1, 23].

Scenarios in the general case, which are considered in the framework of the implementation of an adaptive event-oriented system, may include:

- a list of possible sources corresponding to the request form;
- the number and volume of sessions of requests to sources;
- limitations on the number of requests and search time;
- the initial values of the relative probability of assessing the possibility of having relevant information units on the selection of sources during the information collection.

The BizAgi Process Modeler business process management software platform was used as an instrumental environment for building scenarios and their serialization. A fragment of the graphic semantic model of one of the typical scenarios is presented in Figure 4.

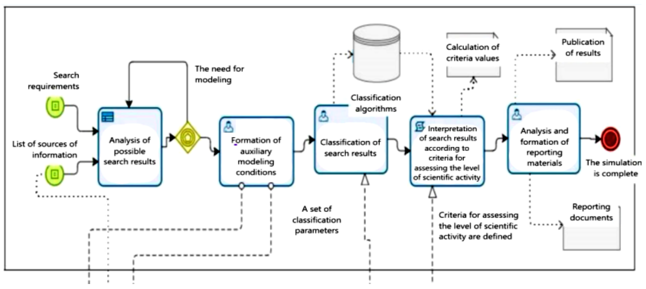

Search requirements
List of sources of information
Analysis of possible search results
The need for modeling
Formation of auxiliary modeling conditions
Classification algorithms
Classification of search results
Calculation of criteria values
Interpretation of search results according to criteria for assessing the level of scientific activity
Publication of results
Analysis and formation of reporting materials
The simulation is complete
A set of classification parameters
Criteria for assessing the level of scientific activity are defined
Reporting documents

*Fig. 4. A fragment of a graphical semantic model of one of the typical scenarios built using the BizAgi Process Modele software platform*

In this case, a unit of information means one separate information file with text content that is considered for its relevance (relevance) to the request. In the absence of additional information, scenario models determine the probability of selecting relevant units of information from all sources considered within this scenario model, which determines the same number of calls to each source for extracting data corresponding to the data retrieval task.

$$P = \{p_1, p_2, p_3, \ldots, p_i, \ldots, p_{n-1}, p_n\}$$

$$\sum_{i=1}^{n} p_i = 1$$

$$p_i = 1/n,\ I = 1, \ldots, n$$

Next, according to the session size set in the scenario, a specific number of data extractions from each source is determined during one session.

$$Q_S = \sum_{i=1}^{n} q_{ij},\ j = 1, \ldots, J_{max}$$

where: $Q_S$ – the total size of the session, i.e., the total number of requests to all sources in one iteration of the method, $q_{ij}$ – session size for the $i$-th source at the $j$-th iteration, $J_{max}$ – the maximum number of iterations in the scenario.

This generates the number of data extractions in each of the sources within the current data collection session. The total size of the session itself is determined by the scenario and can be changed during the execution of the search by the analyst, during the data collection, but, as a rule, it has a constant value. Such a need may arise when, during the collection of information from sources, the number of relevant data units is insufficient and it is necessary to increase the total sample size to achieve certain results.

In the next step, queries are made to the sources of information defined in the scenario. The source group used in this review is defined by the scenario description.

The number and list of sources of information used within the scenario are generally unchanged during the collection of information according to the current request. But, if necessary, to speed up the process of data collection from the sources most relevant to the request, the analyst can reduce the number of sources that are not efficient enough, or identify new additional sources, which will make it possible to increase the efficiency of data collection.

The formation of a group of sources according to the scenario is performed by defining specific microservices that are responsible for extracting information from the relevant sources. Each of the microservices is responsible for sampling information from one specific source. The operation of the microservice, within the framework of this process, is determined by the extraction of a certain amount of data that corresponds to a specific request within the framework of which this microservice performs the extracted information.

The process of interaction of a microservice with an information source consists of the following main steps:

- a request to the source of information;
- extracting a unit of information;
- assessment of the relevance of the extracted unit of information;
- preservation of a relevant unit of information;
- ending the cycle of calls to the source when the number of calls to the source, which is defined within the current session, is reached.

Each of the microservices is focused on extracting information from a specific source. This approach is a consequence of the fact that each of the sources has its specific features:

- structures and forms of preservation of information arrays;
- permissions and restrictions on access to information stored in a certain source;
- the availability of the extraction of this information following the request.

The input data for each selected microservice is:

- number of links to source;
- parameters for the mechanism for assessing the relevance of the received data;
- the degree of compliance.

Under the parameters for the mechanism for evaluating the relevance of the received data, a set of keywords with their weight values can be used as a minimum for this consideration. Then the relevance score $M_l$ for each $l$-th unit of data can look like this:

$$M_l = \sum_{k=1}^{K} H_k W_k$$

$K$ – number of search parameters, $W_k$ – the weight of the $k$-th search parameter according to the search task, $H_k$ – indicator of the presence of the $k$-th search parameter in the document, which has a value of *1* if it is present, or *0* if the $k$-th search parameter is not present in the document.

Then the overall estimate of the availability of relevant data units for each $i$-th source of information in the current $j$-th session can be defined as

$$R_i^j = \sum_{l=1}^{L} M_l$$

$$L = q_{ij}$$

After the session, the results of data collection and assessment of the presence of relevant data in each of the sources are transferred to each of the data collection microservices for their subsequent consolidation and assessment of the results of extracting the necessary data for each of the sources in the current session. Consolidation of received data, within the framework of the system under consideration, means their initial verification by request and bringing them to a single form for the possibility of further processing.

The obtained results are transferred to the database that collects the results of withdrawals. In the future, these data collection results will be used to prepare a report on the task.

Further, depending on the chosen direction of implementation of the algorithm of event-driven control of the microservice architecture, a synchronous or asynchronous approach can be chosen in the management of data collection microservices. Depending on this choice, the moment of event creation is formed, which means the further process of forming the session size for each of the sources in the next step.

With the synchronous algorithm in the management of microservices, the event that determines the need to recalculate the session sizes for each of the sources is formed only after the end of data collection from all sources, regardless of the session sizes for each of them. Thus, the time of occurrence of the event that initiates the process of recalculating the number of requests for each of the sources within the session is determined by the microservice that last finished extracting data from the source corresponding to it. Then, calculations of session sizes for each source are formed based on data received from all sources within the framework of one session after completion of data collection by all microservices, regardless of the time of data collection by each of them.

With an asynchronous algorithm in the management of microservices, the event that determines the moment of recalculating the session sizes for each of the sources is generated at the end of the session by the microservice that first fulfilled the requests within its session first. At the same time, the execution of their sessions by other microservices is not taken into account. Calculations will be made based on the current results of extracting relevant data from all sources. At the same time, it is not taken into account that the selection of data from other sources has not yet been completed. Thus, in this case, for other sources, performance evaluations for the current session are carried out on unfinished sessions. On the one hand, this approach can significantly speed up the process of obtaining relevant data, and on the other hand, it can significantly slow down the adaptation process of determining the most relevant sources in terms of the number of relevant records in sources.

Although this approach takes into account only partial information regarding the assessment of the relevance of all sources, it speeds up the process of adapting the data collection system to the state of the data sources in terms of the availability of records relevant to the request due to the more frequent formation of an event that initiates the recalculation process [11, 21].

Next, the process of the system's operation proceeds to the assessment of the degree of sufficiency of the amount of received data for the possibility of a reasonable assessment of the states of the sources from the point of view of the availability of relevant data. If such data is not enough, it is possible to repeat the previous session to obtain more relevant data, which will be used to recalculation the P vector. When such repeated execution of the session does not improve the results, a situation is possible when it is necessary to change the collection scenario by using a different set of sources for a certain request.

In some cases, the following simplified scheme can be used to quantify the quality of the data collection process from the point of view of obtaining relevant data from relevant sources.

A model is used that allows a partial correspondence between the request and the source of information. This correspondence evaluates to a value in the range [0,1]. The value that evaluates the correspondence between the request and the source of information is formed based on determining the match between the parameters of the request and the characteristics of the information units included in a certain source of information under consideration.

Quantitative evaluation of the correspondence of the *l*-th document (for $l = 1, \ldots, V$) from the *i*-th information source to the *j*-th query parameter during one sample to assess the relevance of the source is calculated as:

$r_{ijl} = 0$ – when the *j*-th parameter is not present in the *l*-th document of the *i*-th source of information,

$r_{ijl} = 1$ – when the *j*-th parameter is present in the *l*-th document of the *i*-th source of information.

So this value, after sampling and evaluation of several units of information from a certain source of information, is averaged in accordance with the number of selected units (documents).

Quantitative assessment of compliance of the *i*-th information source with the *j*-th request parameter determines its partial compliance, it corresponds to the range [0,1] and can be defined as

$$k_{ij} = \frac{1}{V}\sum_{l=1}^{V} r_{ijl}$$

$r_{ijl}$ – is a quantitative assessment of the correspondence of the *l*-th document (for $l = 1, \ldots, V$) from the *i*-th information source to the *j*-th query parameter during one sample to assess the relevance of the source, $V$ – the volume of one single sample from one source of documents must comply with the following limitations:

$$V \ll S_i \text{ for } i = 1, \ldots, n$$

$S_i$ – is the number of information units (documents) in each of the *n* sources of information considered for this request.

Then the assessment of the relevance of the *i*-th source of information will be determined as

$$R_i = \frac{1}{m}\sum_{j=1}^{m} k_{ij}\ v_j$$

where $0 < v_j < 1$, $v_j$ – weight coefficient of the *j*-th parameter of the requested topic, determined by expert evaluations or based on priorities, which can be independently determined by the customer of the information request.

If sufficient relevant data is received to evaluate the data sources, it is checked whether enough data is received according to the request to further prepare the report. When this is the case, a report is prepared according to the request by processing the data extracted by the microservices from the relevant sources and stored in the database of results.

If the amount of received data is sufficient for the possibility of a reasonable assessment of the state of the sources in terms of the availability of relevant data, but when their quantity does not yet meet the needs of the request, the results of requests to the sources for each data source are processed.

Based on these data, an event is formed as a vector of the influence of $G_k$ on the corresponding values of the components of the probability of having relevant units of information from all sources considered within the framework of this scenario model.

$$P_{k+1} = G_k P_k =$$
$$= \{g_{1k}p_{1k}, g_{2k}p_{2k}, g_{3k}p_{3k}, \ldots, g_{ik}p_{ik}, \ldots, g_{n-1k}p_{n-1k}, g_{nk}\, p_{nk}\}$$

In the general case, the value of the influence coefficients can be determined as

$$g_{ik} = F(p_{ik} - p_{ik-1})$$

The choice of one or another type of function *F* can be the result of a separate study.

The effectiveness of the synchronous and asynchronous algorithms largely depends on the quality of the definition of the probability vector of relevant units of information for all sources for the next session of the algorithm. Regardless of the type of methods described above, in the simplest case, it is possible to determine that the vector calculation of the probability of the presence of relevant units of information for all sources can be based on the use of both the previous values of the probability estimates for several previous sessions of the search process and the degree of consideration of these values for next session [11].

The basis for these estimates is the assessment of the change in the amount of relevant data by session from the previous session to the next.

$D^S_i = (R^S_i - R^{S-1}_i)/ R^{S-1}_i$ – the relative change in the amount of relevant data between sessions *S* and *S-1*

The estimate of the amount of relevant data in the next *S+1* session can be defined as

$$\check{R}^{S+1}_i = R^S_i + D_s$$

An estimate of the value of the number of data extractions for the *i*-th data source for the next session can be calculated as follows:

$$K^{S+1}_i = F(\check{R}^{S+1}_i)$$

where *F* – the evaluation function, which determines the implementation of the event of a change in the value of the amount of relevant data between consecutive sessions for each of the sources.

It is possible to evaluate the effectiveness of retrieving relevant data for the current and previous sessions for all sources as

$$E^{S-1}_i = \frac{R^{S-1}_i}{K^{S-1}_i},\ E^S_i = \frac{R^S_i}{K^S_i}$$

where *i=1, ..., N.*

With

$$K^0_i = const = \frac{Q}{N},\ R^0_i = 0$$

Based on the obtained evaluations of the extraction of relevant data for the current and previous sessions for each of the sources, it is possible to evaluate the change in efficiency as a factor that can influence the determination of the decrease or increase in the values of the total amount of data $K^{S+1}_i$ received from the *i*-th source for the *S*-th session. Then you can determine what.

$$D^S_i = E^S_i - E^{S-1}_i$$

At the same time, the change in efficiency for each session should be affected in a certain way with a certain coefficient of value, which should reduce its value for earlier sessions. This can be defined as the effect of partially reducing the influence of previous sessions on subsequent ones.

The total effect on $K^{S+1}_i$ can be defined as

$$Z^{S+1}_i = \sum_{j=1}^{S} D^j_i v^{j-S+1}$$

where *v* – is the coefficient of influence of the results of previous sessions.

At the same time, not the entire previous history of sessions can be analyzed to determine the impact, but only the value of the effectiveness of several previous sessions, then

$$Z^{S+1}_i = \sum_{j=h}^{S} D^j_i v^{j-S+1}$$

where *h* – determines the number of previous sessions taken into account when taking into account the influence of previous sessions.

In this consideration, the total amount of data received from the *i*-th source for the *S+1*-th session in a simpler fit can be defined as

$$K^{S+1}_i = K^S_i (1+Z^{S+1}_i)$$

After that, the obtained results are normalized according to the calculated sum of $K^{S+1}_i$ grades throughout the session.

To study the possibilities and effectiveness of the described synchronous and asynchronous algorithms for the implementation of event-driven management of the selection of sources during the execution of a request for the collection of relevant data, there is a study of the effectiveness of the process depending on such parameters as:

- first, the number of previous sessions, the results of which are taken into account when taking into account the influence of previous sessions *h*;
- secondly, the coefficient of influence of the results of previous sessions *v*.

Analysis of the effectiveness of synchronous and asynchronous algorithms for the implementation of event-driven management of source selection shows a significant dependence of their effectiveness on the values of these parameters and their combinations.

## 3. Comparative testing

To analyse and compare synchronous and asynchronous control methods, the described algorithm was tested on the generated test models of data sources [25]. For each test, 10 test data source models of 10,000 records each with a percentage of relevant records between 1% and 10% were generated and used.

Without reducing the degree of generality, an even distribution of records relevant to the respective queries in each of the source models was used to fill the test source models. Thus, the number of records relevant to the query in the source models was variable from 100 to 1000, corresponding to a percentage of relevant records from 1% to 10%. At the same time, each of the test models was implemented as a separate file of 1000 records, which made it possible to use these source model multiple times to analyze the influence of the number of previous sessions and the coefficient of influence of the results of previous sessions on the probability of obtaining relevant data from each of the source models.

For ease of implementation of testing of synchronous and asynchronous algorithms, relevance was defined as a complete match of the symbols included in the request with the symbols of records in the source model. This makes it possible, with little time for data processing when testing algorithms with various parameters, to determine the features and capabilities of synchronous and asynchronous algorithms in managing data collection quite fully and comprehensively.

During testing, various combinations of *h* and *v* values were considered. Below are those combinations of values that gave the most characteristic results.

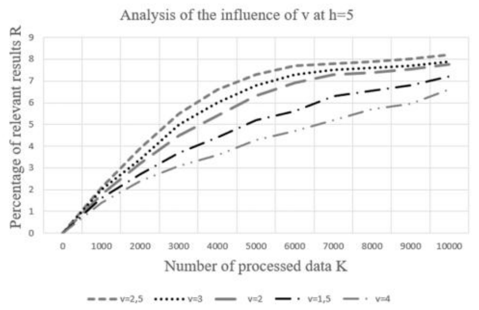


*Fig. 5. Synchronous model. Analysis of the influence of v at h = 5*

Figure 5 shows the obtained test results, showing the dependence of the percentage of relevant results obtained on the influence coefficient of the results of previous sessions *v.* At the same time, the value of the number of previous sessions, the results of which are taken into account when taking into account the influence of previous sessions *h* was unchanged and equal to 5. The best results were achieved when the value of the coefficient was equal to 2.5. Further refinements showed that this value is closer to 2.7.

Figure 6 shows the obtained results of the study, showing the dependence of the percentage of relevant results obtained on the number of previous sessions, the results of which are taken into account when taking into account the influence of previous sessions *h*. The best results were obtained when the value of the number of previous sessions taken into account was *h* = 5. As the value was further increased (*h = 6, 7, …*) the improvement in results was practically negligible to account for its improvement. At the same time, the coefficient of influence of previous sessions v was unchanged and equal to 2.5, according to the best results.

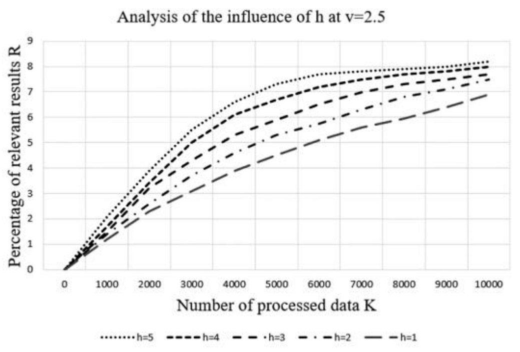


*Fig. 6. Synchronous model. Analysis of the influence of h at v = 2.5*

Similar results for the values of $h = 5$ and $v = 2.5$ were obtained for the asynchronous algorithm, which is shown in Figures 7 and 8. Such values were obtained as the most qualitatively relevant to the testing task for the vast majority of conducted studies of the method.

However, it should be noted that the results of the synchronous and asynchronous microservice management algorithms in the event-oriented microservice system, which is described, have significant differences from the point of view of the process of manifestation of the adaptive properties of the system.

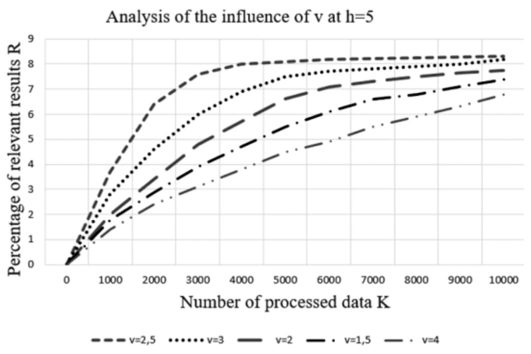


*Fig. 7. Asynchronous model. Analysis of the influence of v at h = 5*

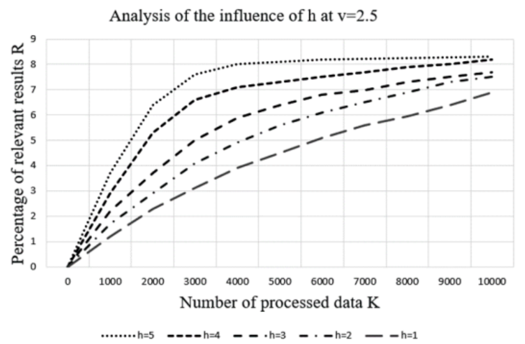


*Fig. 8. Asynchronous model. Analysis of the influence of h at v = 2.5*

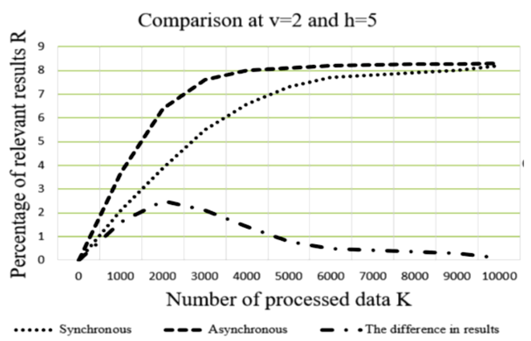


*Fig. 9. Comparative analysis of synchronous and asynchronous algorithms of an adaptive event-driven microservice system*

The results of the analysis shown in Figure 9 show that, depending on the size of the data sample, the methods behave differently.

An event-driven algorithm built based on an asynchronous algorithm proves to be much more effective with a smaller number of attempts to search for relevant data due to a higher speed of adaptation. This may be a manifestation of the more frequent occurrence of events that correct the adaptation procedures in connection with the end of the session based on the results of the first of the microservices, which finished the selection without waiting for the end of the selection of data from other sources.

But with an increase in the number of processed data selected from sources, the adaptive event-driven algorithm built based on the synchronous method becomes no less effective and even more stable from the point of view of the constant increase in the percentage of selected relevant data units. This, in turn, can be considered as a manifestation of a more complete assessment of information regarding the availability of relevant data from all sources. Practically, with a large number of processed results, the difference between the percentages of relevant results for synchronous and asynchronous algorithms becomes approximately the same.

## 4. Conclusions

The developed event-driven architecture of the system for processing large data flows makes it possible to collect information from various forms of storage and composition of sources depending on the tasks defined by the user. The composition of information sources in this system can be expanded without interfering with other components of the system. Each microservice configured for one specific source or several homogeneous sources of information can provide only partial data, which is supplemented by other sources.

In the course of its work, the system in real time adjusts to the user's request for information in such a way that the data is taken mainly from those sources that meet the requirements of the request and can satisfy them. At the same time, source selection procedures based on a linear stochastic automaton [26] can be used, which generates requests to sources that contain the maximum amount of data relevant to the request. The described architecture of the software system makes it possible to manage parallel data processing with significant independence of the system programs from the scale of the system.

An important advantage of the event-driven architecture in the system for processing large data flows is also the possibility of adapting the data collection scenario due to the selection of sources by the quantitative assessment of the relevance of information obtained during the execution of user requests.

The considered synchronous and asynchronous methods in the implementation of event-oriented architecture can be used in different software systems depending on their purpose. An asynchronous approach can have significant advantages in implementing operational real-time search and data collection systems. A synchronous approach may be more attractive when developing information and analytical systems that cyclically or continuously process data flows from disparate sources, the volume of which is constantly increasing.

## References

[1] Akhtanov S., Turlykozhayeva D., Ussipov N., Ibraimov M., Zhanabaev Z.: Centre including eccentricity algorithm for complex networks. Electronics Letters 58(7), 2022, 283–285.

[2] Al-Masri E.: Enhancing the Microservices Architecture for the Internet of Things. IEEE International Conference on Big Data (Big Data). USA, WA, Seattle, 2018, 5119–5125.

[3] Azarov O. et al.: Means of analyzing parameters of speech signal transmission and reproduction. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska 14(2), 2024, 11–16.

[4] Azarova A. O. et al.: Information technologies for assessing the quality of IT-specialties graduates' training of university by means of fuzzy

logic and neural networks. International Journal of Electronics and Telecommunications 66(3), 2020, 411–416.
[5] Belnar A.: Building Event-Driven Microservices: Leveraging Organizational Data at Scale. O'Reilly Media, USA 2020.
[6] Bisikalo O. et al.: Parameterization of the Stochastic Model for Evaluating Variable Small Data in the Shannon Entropy Basis. Entropy 25(2), 2023, 184.
[7] Buyya R.: Big Data. Principles and Paradigms. Elsevier, 2016.
[8] Chris R.: Microservices. Development and refactoring patterns. Peter, 2019, 544.
[9] Davis A.: Bootstrapping Microservices with Docker, Kubernetes, and Terraform: A project-based guide. Manning, Shelter Island 2021.
[10] Dinesh R.: Hands-On Microservices – Monitoring and Testing. Hands-On Microservices – Monitoring and Testing: A performance engineer's guide to the continuous testing and monitoring of microservices. Packt Publishing. 2018.
[11] Erl T.: Big Data Fundamentals. Concepts, Drivers & Techniques. Prentice Hall, 2016.
[12] Ford N., Parsons R., Kua P.: Building Evolutionary Architectures: Support Constant Change. O'Reilly Media, 2017.
[13] Ghiya P.: Typescript Microservices: Build, deploy, and secure microservices using TypeScript combined with Node.js. Packt, Birmingham 2018.
[14] Gorelik A.: The Enterprise Big Data Lake: Delivering the Promise of Big Data and Data Science. O'Reilly, 2019.
[15] Koval O. V. et al.: Evaluating the Quality of Modeling the Scenario of Information Analysis on a Branched Network. Modern information protection. DUT 3(39), 2019, 70–76.
[16] Koval O. V. et al.: Improving the Efficiency of Typical Scenarios of Analytical Activities. CEUR Workshop Proceedings 3241, 2021, 123–132.
[17] Koval O. V. et al.: Refining the typical scenarios by additional factors. Mathematical and computer modeling. Series: Technical sciences 1(20), 2019, 68–78.
[18] Kuzminykh V. O. et al.: Data collection for analytical activities using adaptive micro-service architecture. Registration, storage and processing of data 23(1), 2021, 7–79.
[19] Kuzminykh V., Xu B.: The influence of current results in an event-oriented data collection system. Zviazok 3(169), 2024, 18–22.
[20] Mamyrbayev O., Toleu A., Tolegen G., Mekebayev N.: Neural architectures for gender detection and speaker identification. Cogent Engineering 7, 2020, 1727168, 1–13.
[21] Newman S.: Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, 2015.
[22] Rocha H. F. O.: Practical Event-Driven Microservices Architecture: Building Sustainable and Highly Scalable Event-Driven Microservices. Apress, 2021.
[23] Shuiskov A.: Building Microservices with Go: Develop seamless, efficient, and robust microservices with Go. Packt Publishing, 2022.
[24] Simon P.: Too Big to Ignore: The Business Case for Big Data. Wiley, 2019.
[25] Turlykozhayeva, D. et al.: Routing Algorithm for Software Defined Network Based on Boxcovering Algorithm. 10th International Conference on Wireless Networks and Mobile Communications (WINCOM), 2023, 1–5.
[26] Wolff E.: Microservices, Flexible Software Architecture. Addison-Wesley, Boston 2016.
[27] Zgurovsky M. Z., Zaychenko Y. P.: Big Data: Conceptual Analysis and Applications. Springer, 2020.
[28] Zhang H., Li S., Jia Z, Zhong C., Zhang C.: Microservice Architecture in Reality: An Industrial Inquiry. IEEE International Conference on Software Architecture (ICSA), Germany, Hamburg, 2019, 51–60.

**Ph.D. Valeriy Kuzminykh**
e-mail: vakuz0202@gmail.com

Ph.D. of Engineering Sciences, associate professor, Department of Software Engineering in Energy, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Author of more than 70 scientific publications, of which 18 are in scient metric databases Scopus.

https://orcid.org/0000-0002-8258-0816

**Prof. Oleksandr Koval**
e-mail: avkoval@gmail.com

Doctor of Engineering Sciences, professor, Department of Software Engineering in Energy, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Author of more than 100 publications, including 3 textbooks, and more than 50 scientific articles in professional journals, of which 20 are in scient metric databases Scopus and Web of Science.

https://orcid.org/0000-0003-0991-6405

**Prof. Yevhen Havrylko**
e-mail: gev.1964@ukr.net

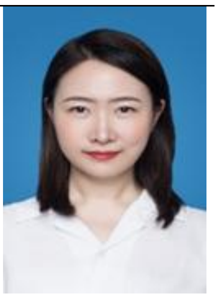
Doctor of Engineering Sciences, professor, Department of Software Engineering in Energy, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Author of more than 50 publications, including 3 textbooks, 3 patents for inventions, and more than 35 scientific articles in professional journals, of which 17 are in scient metric databases Scopus and Web of Science.

https://orcid.org/0000-0001-9437-3964

**M.Sc. Beibei Xu**
e-mail: xubeibei1987@163.com

Ph.D. student, Department of Software Engineering in Energy, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Author of 7 scientific publications which are in scient metric databases Scopus.

https://orcid.org/0000-0003-1430-5334



**Prof. Iryna Yepifanova**
e-mail: yepifanova@vntu.edu.ua

Doctor of Economic Sciences, professor, Vice-rector by science work, Faculty of Management and Information Security of Vinnytsia National Technical University, academician of the Academy of Economic Sciences of Ukraine
Scientific interests: financial support of innovative activities of domestic enterprises, enterprise potential, competitiveness, personnel management, digital economy, energy saving

https://orcid.org/0000-0002-0391-9026

**M.Sc. Shiwei Zhu**
e-mail: zhusw@sdas.org

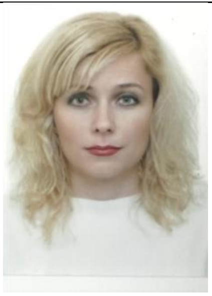
Master of computer science, Ph.D. student, Department of Software Engineering in Energy, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine.
Author of 16 scientific publications.

https://orcid.org/0000-0002-6651-8449

**Ph.D. Nataliia Bieliaieva**
e-mail: i-natali-@ukr.net

Doctor of Philosophy, assistant, Faculty of Technology and Design, Dragomanov Ukrainian State University, Ukraine, Kyiv.
Scientific interests: pedagogical technologies, financial support of innovative activities of domestic enterprises, enterprise potential, competitiveness, personnel management,

https://orcid.org/0009-0003-2987-8423



**Ph.D. Bakhyt Yeraliyeva**
e-mail: yeraliyevabakhyt81@gmail.com

Senior lecturer of the Information Systems Department, Faculty of Information Technology, M. Kh. Dulaty Taraz Regional University, Taraz, Kazakhstan.
Research interests: fiber optic technologies, information systems, Internet of Things and blockchain technologies.

https://orcid.org/0000-0002-8680-7694