Towards Simplifying the Use of Self-Services

Kaspars Kalnins[1] and Marite Kirikova[1]

[1] *Institute of Applied Computer Systems, Riga Technical University, 6A Ķīpsalas Street, Riga, LV-1048, Latvia*

Abstract
While largely applied to different platforms, self-service intelligence (or self-service analytics) still faces challenges in its practical usage. As the amount of data and types of analytics has increased, a new requirement emerges to store existing analytics results so that they can be accumulated and reused. Therefore, it is necessary to develop a method to provide a process where the analytics results obtained from the platform are automatically saved to the database so that users without technical knowledge can implement this with a low-code and self-service BI approach. In this work, various literature sources are studied, resulting in a feature list for implementing a low-code approach; and a process model is developed for the method. The paper focuses on Oracle BI as a platform that allows users to analyse data of different nature.

Keywords
Oracle BI, low-code, self-service, self-service intelligence

1. Introduction

People with the skills and abilities to process data quickly and efficiently are increasingly in demand in many sectors [1]. Many industries need to start thinking about creating a data culture in their companies so that everyone understands that data is an asset to the company [2]. This asset serves as a foundation for further development because data can be used to create information and knowledge in a particular context [2]. It is also essential to understand that analytical data must be available to be served to managers as quickly as possible to make decisions [3]. As the amount of data being processed worldwide grows and is projected to reach 180 ZB by 2025 [4], it is necessary to understand what the company should do with it, whether to store it, process it, delete it, etc. Considering all this, Business Intelligence (BI) should be an integral part of companies today [1]. BI tools can structure and transform data into a business asset. While the amount of data increases, it is often not the data itself that is important to the industry but the information that BI produces. Therefore, to save resources, companies do not store data for the long term but only as long as the information is extracted using BI tools. After the data has been processed with the BI tool, the data is deleted, and analytical information is stored, which serves as a

Baltic DB&IS Conference Forum and Doctoral Consortium 2024
kaspars.kalnins_2@edu.rtu.lv (K.Kalnins); marite.kirikova@rtu.lv (M.Kirikova)
0000-0002-1678-9523 (M.Kirikova)

source of knowledge for future decision-making. It is, therefore, more important for companies today to focus on the ability to process and store the results of BI solutions [4].

Self-service intelligence has been seen as a solution to this problem [5]. However, the challenges in the use of this approach have also been reported [5]. Therefore, this paper focuses on applying self-service intelligence in business intelligence. To do this, we use the Oracle BI platform, which offers a wide variety of ready-made self-service tools [6], and we attempt to define a method for extending Oracle BI functionality. Specifically, the goal of this research is to define, using a low-code and self-service BI approach, a method that would allow a user without technical knowledge to save analytics results in an automated way. Extrapolation of findings to other BI platforms or tools is beyond this paper's scope.

The paper is organised as follows. In section 2 we describe, in more detail, the problem addressed in the paper and formulate the research questions. Section 3 illustrates the method of literature analysis while the literature analysis results are presented in Section 4. The brief concluding remarks are available in Section 5.

2. Knowledge Debt in Self-Service Business Intelligence

Companies need experts to use BI tools, but not all companies have them. Many companies, therefore, choose to proceed without these tools [3]. BI contains such components as a data warehouse, data extraction, pre-processing, and result output system [7]. Even with an automated BI output system, users need additional methods to process the data after the first processing. The final analytical reports are produced by end-users with a background and position outside the IT sector [8]. Typically, BI staff is divided into casual and power users. Casual users are the employees of the company who need the results of the data analysis to make decisions. Power users are experts who perform technical operations to obtain the results of the data analysis. In a typical process, the casual user requests the power user to develop the analytical solution. However, as the quantity of data increases and the need to view the data from different perspectives increases, companies need more technical resources to develop the analytics. As a solution to this problem, a Self-Service Business Intelligence (self-service BI) trend has developed, where casual users should be able to create analytics solutions without the involvement of technical staff. However, since BI development requires specialised knowledge, casual users must possess such knowledge. Because of technical difficulties, BI tools should be as little as possible based on technical expertise for implementing self-service BI in the company [5].

In BI, data is structured and stored in a data warehouse. Next, the data is structured into models using analytical tools, and pre-processing is performed. The final step is producing the analysis output through reports and statements (Figure 1. a). It should be noted here that the power user maintains the Data Warehouse and the analytical tools, while the casual user handles the final output. According to the self-service BI, the aim is for the casual user to be able to use the analytical tools for data management (Figure 1. b). Studies have shown that companies are more productive by adopting this practice [1].

One of the self-service BI tools available today is Oracle Analytics Server (Oracle BI). Oracle BI gives users the tools to perform analytics [9]. Even if Oracle BI with self-service solutions allows the development of analytical solutions using a low-code or no-code approach, the system does not allow the possibility for a casual user to save analytics results in an automated way for further analytics creation. The user has to manually save the analytical result data from the system daily, as Oracle BI does not offer such functionality. Manual operations are appropriate if they need to be performed occasionally. However, with the increasing amount of data processing, the number of such manual activities also increases, and as a result, the company loses staff resources on manual activities.

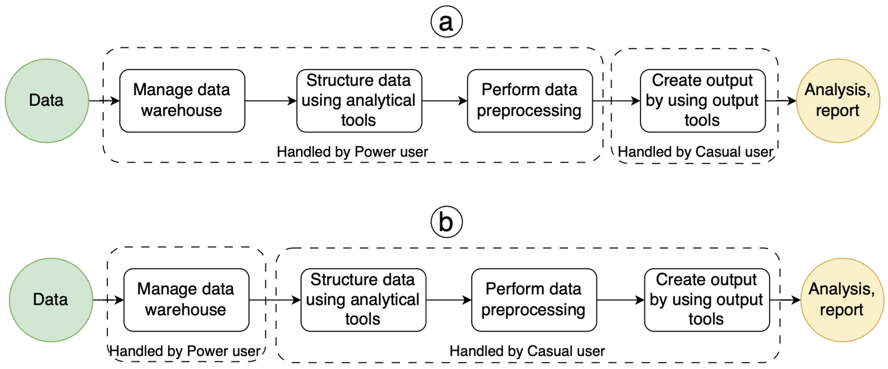

a
Data
Manage data warehouse
Structure data using analytical tools
Perform data preprocessing
Handled by Power user
Create output by using output tools
Handled by Casual user
Analysis, report
b
Data
Manage data warehouse
Handled by Power user
Structure data using analytical tools
Perform data preprocessing
Create output by using output tools
Handled by Casual user
Analysis, report

Figure 1: Data processing flow in BI: a) using regular approach; b) using self-service BI approach.

It is essential for BI tools to provide not only data analysis but also the transfer of results via service-oriented architecture (SOA) or microservice architecture [10]. It is, therefore, essential to explore the possibilities of enhancing the Oracle BI functionality so that the casual user can organise automated storage of analytics results using a low-code approach, i.e., to implement solutions using as many graphical tools as possible, without additional coding. It is also necessary to investigate the process of saving the data of such analytical results so that the user can use a self-service BI approach, i.e., to minimise usage of technical knowledge. Therefore, the hypothesis is that by combining Oracle BI with web server and database systems into service, a method can be created to store analytical results with low-code and self-service BI approaches. The following two questions are further researched: (RQ1) "What Oracle BI low-code features are available for development?" and (RQ2) "What is the Oracle BI self-service process to enhance functionality for storing data of analysis?".

The answers to these questions would help to identify the methods for providing more comfortable self-service BI solutions. As BI platforms differ, at this stage of research we have focused on one platform only.

3. The Analysis of Related Works

To carry out the research, the literature review method was used based on Levy and Ellis [11]. Initially, background information was gathered, and research questions were stated (Section 2 of this paper). The review process was then accomplished in three stages [11]: input, processing, and output. During the input stage, literature was searched and selected from various sources. The information in the selected sources was comprehended, applied, analysed, synthesised, and evaluated during processing. In the output stage (Section 4 of this paper), research findings were synthesised into new knowledge [11].

Initially, the keyword list was determined according to the achievable goal: Big data analytics, SQL—to look for articles with data processing; OBIEE—the acronym for the Oracle BI platform; low-code, self-service BI—to find solutions for casual users; microservices, SOA—to find articles with service solutions; PHP low code—to look for web server solutions. Various combinations of keywords were used to search for the literature in electronic resources such as Scopus and Google Scholar. The results from search engines were used to look for documents in the electronic databases. Also, the year of publication was considered to understand if this literature source is up-to-date. By analysing the first search results with keywords OBIEE, it was understood that articles before 2017 were related to the older Oracle BI platform's technical specifications, which are irrelevant to technological solutions nowadays. Therefore, all articles published before 2017 were filtered off. For each found related work, an abstract was read to see if it was related to the achievable goal. If the abstract was unrelated, then the article was excluded. For each found-related work, a forward search was performed. If another literature source had cited the article and if the abstract was relevant to the topic, then the source was included in the list. By forward search, nine articles were found. Similarly, like a forward search, a backwards search was performed to see which literature sources the paper used. If the topic was relevant, then it could be added to the list. By backwards search, no articles were found. Afterwards, literature sources were excluded if they were found unrelated to the research topic by reading the whole article.

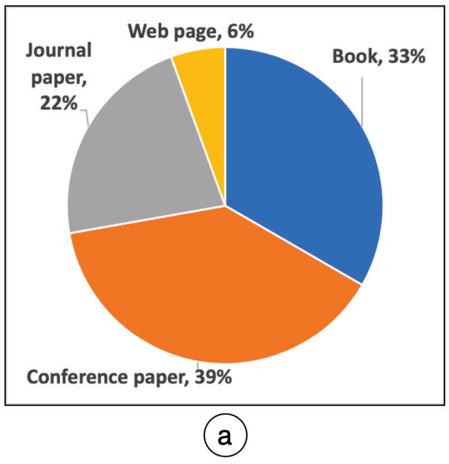
Table 1 shows the distribution of retrieved articles by publisher.

Table 1
Article distribution by publisher

Publisher	Search results	Selected literature
ACM Digital Library	4	3
Australasian Journal of Information Systems	1	1
CSIMQ	1	1
IEEE Xplore Digital Library	3	2
IGI Global	1	1
John Wiley & Sons	1	1
O’Reilly	1	1
Pearson Education	1	1
Science Direct	1	1
Springer Link	8	4
Taylor & Francis	1	1
Oracle	1	1
Total	24	18

In Table 1, we can see that, after the search, 24 articles were identified, but only 18 were relevant to the topic after their deeper analysis. 39% of the articles are conference papers

(Figure 2. a). Table 2 shows the distribution of literature by type, and Figure 2. b shows its distribution by years of publishing.



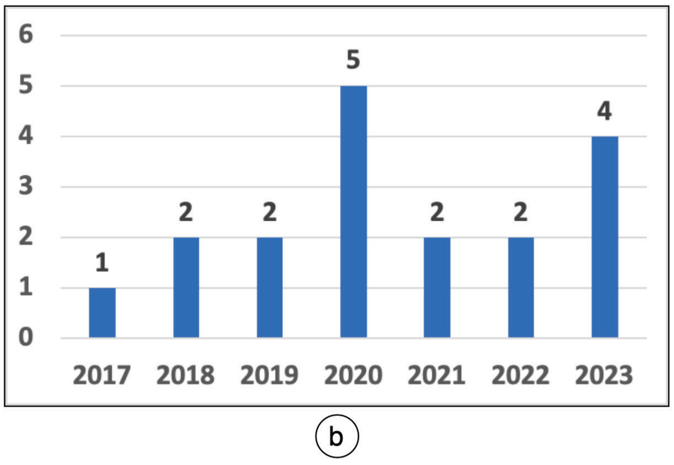


Figure 2: Literature distribution by: a) type in percentage; b) by year.

Table 2
Literature distribution by type

Type of source	Number of sources
Book	6
Conference paper	7
Journal paper	4
Web page	1

In literature processing, information synthesis was carried out by extracting relevant knowledge from the articles and then combining it in a substantive way to answer the research questions. The research results obtained after the synthesis are described and discussed in Section 4.

4. Research Results

The answers to the stated research questions are organised in Sections 4.1 and 4.2; and the results are discussed in Section 4.3.

4.1. Oracle BI Low-Code Features

Based on the [12], [13], and [14], for a low-code approach to be implemented in a BI system (platform or tool), the system must contain several features. Table 3 lists the required essential features and evaluates whether Oracle BI contains them in its functional description [6], [9]. Additionally, the above-mentioned literature sources emphasise that low-code platforms (in our case, the self-service BI platform) need excellent system performance, real-time behaviour, high data processing, and code automation.

Table 3
Oracle BI low-code features

No	Features from [12]-[14]	Oracle BI platform (system) characteristics according to [6] and [9]
1.	Requirement modelling support	The Oracle BI platform does not contain functionality that enables requirements management.
2.	Visual development tools	Oracle BI supports visual designers with drag-and-drop properties, and the Deliver functionality provides a fillable form with which it will be possible to adjust the process and use advanced coding components to obtain non-standard solutions.
3.	Reusability support	Oracle BI supports SOA and Microservices integrations, which, according to the [15], means that functionalities are reusable.
4.	Data source specification management	The Oracle BI system allows connection to different sources and model data structures.
5.	Interoperability support	The Oracle BI system allows connections to external systems for both sending and receiving data.
6.	Business logic specification mechanism	The Oracle BI system does not have built-in functionality to manage business rules.
7.	Development automation features	The Oracle BI system allows agent-based automation processes.
8.	Collaborative development support	Oracle BI does offer collaboration possibilities for developing a single solution.
9.	Artificial intelligence	There is no possibility for AI directly influencing Oracle BI system processes or solutions.
10.	Testing and verification support	It is possible to organise tests in the Oracle BI platform.
11.	Deployment support	Oracle Middleware manages application deployments.
12.	Security support	The Oracle BI system fully secures both the apps and the platform.
13.	Lifecycle management features	The Oracle BI system does not provide functionality for the historical development of solutions.
14.	Analysis environment	The Oracle BI system provides both analysis and reporting functionality.
15.	Extensibility	Oracle BI enables connections to other extensions using SOA or Microservices principles.
16.	Scalability	The Oracle BI system allows control of connections, traffic, and server load.

4.2. Oracle BI Self-Service Process

Oracle BI functionality allows the user to invoke software agents to send data. The agents can call SOA or HTTP request processes. As SOA is usually integrated into a specific business process, then, in such cases, the SOA approach cannot be used to create different non-standard processes for data storage, as the results of each analysis may be relevant to another business process in the enterprise [15]. To gain flexibility and use the self-service BI approach, it is necessary to use the microservice approach, which in Oracle BI can be done using the HTTP request functionality [6].

Since Oracle BI agents call an event, it is the reason to use an event-driven microservices approach [15], [16]. According to the challenges of using BI tools [8] and implementing self-service BI [1], [5] in companies, microservice as the solution can be adjusted to meet the requirements of employees. The reusability of service helps company managers get to needed data faster and make decisions [3], [7] as they will not have to wait long for solution development. This approach gives the flexibility to scale the data structure and volume and be ready to increase data processing volumes in the future [4]. By introducing low-code principles, it must be respected that requirements will grow over time. The possibility of extending functionality must be foreseen [17], and microservices are more flexible for such changes [15].

For designing a self-service support process that makes it easier for a user to utilise the self-service approach, it is necessary to understand who or what is initiating the process and what value is derived from it [18]. The challenge for the BI system is storing the analytics results in a database. In this case, the process is initiated by the event of the availability of the analytical results, but the objective is to have these results stored in the database by a specified date, which is valuable for the reuse of these analytical results.

In the proposed process (Figure 3), the first activity will be sending analytics results. Since Oracle BI will transfer data using the agent functionality, a casual user must add a table name to the agent parameter [6]. As the service will be reused for different analytical reports, the user must define which database table to use to store each analysis. Because Oracle BI will initiate HTTP requests, we need a web server to do data pre-processing tasks [19]. During the data pre-processing activity, the data must be prepared for storage in the database. Once the data is ready, it gets stored in the database; therefore, the last activity will be storing the analytics results in the database.

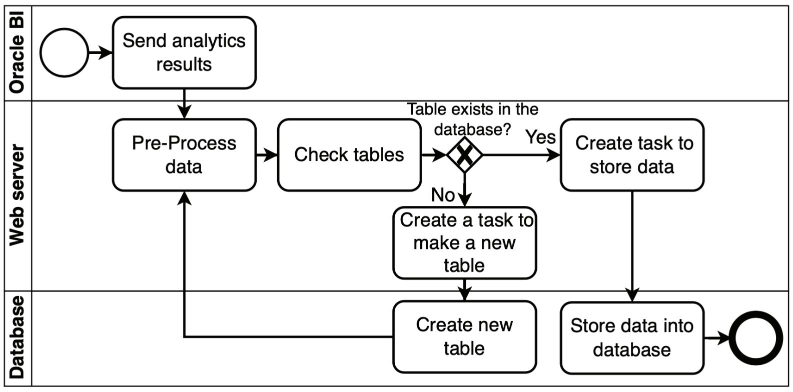


Figure 3: Process to save analytical data into the database.

However, since we need to make this functionality (that provides needed comfort for the end user) repeatable for many instances, it is necessary to foresee that each developed data analytics task needs its own data table created in a database to store the data. Therefore, in the process, we have to consider two cases: one when a new table has to be created in the database and the second when the table is already available [20].

Based on the analysis of related works discussed above, a simplified process model to store analytics results in the database was created, as shown in Figure 3. In the web server part of this model, three additional activities were added, checking the table name to see if such in the database exists; if not, then instructing the database to create a table or to add new data if the table exists. These activities were added because the web server communicates with the database [19]. Therefore, the functions and their detailed activities should be defined in the web server part. Also, additional functionalities can be specified here if they are required in the future [17].

4.3. Discussion

Evaluating the research results on Oracle BI low-code features, Table 3 shows that almost all parameters are satisfactory for extending the functionality of the Oracle BI system following low-code principles. Features such as requirement modelling support, business logic specification mechanism, and lifecycle management, which Oracle BI did not support, depend more on the company if they have integrated such features. The absence of these features may need to be addressed later by understanding that these are not technical features that prevent implementing solutions at this part. Artificial intelligence features, which also are not supported directly by Oracle BI, can be implemented into the process part, outside of Oracle BI, as additional functionality managed by the web server if needed, as it was researched previously for the Oracle BI self-service process.

The research helped to develop a process model for Oracle BI self-service. The model's main feature is that casual users can use the functionalities in an automated way through the microservice. The only task for casual users would be configuring the agent, where the table name as a parameter should be added, and pointing to which database table process should store analytical data. It should also be noted that the established process can be extended with additional functionalities in the web server activity field and additional parameters in the agent if required. Therefore, the possibility of adding different parameters to an agent in combination with the possibility of a web server to define new functions allows the development of various kinds of logic in the backend. Using such a combination allows the creation of a method to extend the Oracle BI platform's functionality by keeping self-service BI and a low-code approach. As a result, the user will not have to code anything. The user can invoke changes by adding parameters to the agent to trigger appropriate web server functions. Even though the process has been modelled, it is currently impossible to say what each activity will have as inputs and outputs because the literature does not cover exact solutions for systems to communicate with each other.

The research results allowed the development of a method to create a service that will extend the Oracle BI functionality to store analytics results in a database using an automated approach. Still, the results indicate that several features are unavailable in Oracle BI, which can cause problems in implementing a low-code approach. Also, how the systems will communicate must be clarified to close the gap in the process. Therefore, to solve these problems, it is necessary to investigate what additional functionalities need to be implemented in the enterprise to provide the missing features and extract the process activities so that the input and output for each activity can be defined precisely.

5. Conclusion

The objective of the current research was to find a method to extend Oracle BI functionality to allow a casual user without technical knowledge to save analytics results in an automated way using a low-code and self-service BI approach. To carry out the research, two research questions were stated: "What Oracle BI low-code features are available for development?" (RQ1) and "What is the Oracle BI self-service process to enhance functionality for storing analysis data?" (RQ2).

As a result, a list of existing and missing Oracle BI low-code features was acquired to answer RQ1. The available features of the system confirm that a low-code approach can be achieved by extending the system with appropriate functionality. However, as discussed in Section 4, the features that need to be added can be implemented with additional external resources if needed. The process model resulting from the research (RQ2) shows all the necessary activities to be performed for the results of Oracle BI analytics to be stored in the database. The process is expected to be run using an event-driven microservice approach. The process model is designed to be used by the casual user with a low-code and self-service BI approach, and it is also open to expanding its functionality in the future in case of new requirements. By combining agent and web server features, the method was developed to extend Oracle BI functionality, which allows casual users to keep working by applying self-service BI and a low-code approach. During the development of the process model, a gap was also identified, as the proposed method is too general to define inputs and outputs for each activity.

Further research needs to be done to find solutions for the missing feature of the low-code approach in Oracle BI, as well as to investigate how the systems in this platform communicate with each other to be able to define inputs and outputs for each process activity. It is also intended to check whether the method presented in this paper can be extrapolated to other platforms to define general requirements for similar self-service support in BI platforms.

References

[1] M. Pałys, A. Pałys, Benefits and Challenges of Self-Service Business Intelligence Implementation. Procedia Comput Sci 225, 795–803 (2023). doi: 10.1016/j.procs.2023.10.066.

[2] L. Veldkamp, Valuing Data as an Asset, Rev Financ 27(5), 1545–1562 (2023). doi: 10.1093/rof/rfac073.

[3] A. Mosbah, M. A. M. Ali, N. M. Tahir, Empowering Small and Medium Enterprises with Data Analytics for Enhanced Competitiveness, in: Proceedings – 13th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2023, pp. 338–342 (2023). doi: 10.1109/ICCSCE58721.2023.10237151.

[4] K. Vassakis, E. Petrakis, I. Kopanakis, Big Data Analytics: Applications, Prospects and Challenges, in: Skourletopoulos, G., Mastorakis, G., Mavromoustakis, C. X., Dobre, C., Pallis, E. (eds), pp. 3–20. Springer International Publishing (2018). doi: 10.1007/978-3-319-67925-9_1.

[5] C. Lennerholt, J. V. Laere, E. Söderström, User-Related Challenges of Self-Service Business Intelligence, Information Systems Management 38(4), 309–323 (2021). doi: 10.1080/10580530.2020.1814458.

[6] Oracle, Visualizing Data in Oracle Analytics Server. 2023. URL: https://docs.oracle.com/en/middleware/bi/analytics-server/user-oas/oracle-analytics-server.html#GUID-6A05572D-8437-40D1-9A4F-44BC37CE7C1C

[7] M. R. Llave, A Review of Business Intelligence and Analytics in Small and Medium-Sized Enterprises, International Journal of Business Intelligence Research 10(1), 19–41 (2019). doi: 10.4018/IJBIR.2019010102.

[8] M. Wee, H. Scheepers, X. Tian, Understanding the Processes of how Small and Medium Enterprises derive Value from Business Intelligence and Analytics, Australasian Journal of Information Systems 26, (2022). doi: 10.3127/ajis.v26i0.2969.

[9] R. Abellera, L. Bulusu, Oracle Business Intelligence with Machine Learning: Artificial Intelligence Techniques in OBIEE for Actionable BI, 1st ed., Apress Berkeley, CA, 2017. doi: 10.1007/978-1-4842-3255-2.

[10] D. P. Wangoo, Intelligent Software Mining with Business Intelligence Tools for Automation of Micro services in SOA: A Use Case for Analytics, in: 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 98–101. IEEE (2020). doi: 10.23919/INDIACom49435.2020.9083682.

[11] Y. Levy, T. J. Ellis, A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research. Informing Science, The International Journal of an Emerging Transdiscipline 9, 181–212 (2006). doi: 10.28945/479.

[12] K. Rokis, M. Kirikova, Exploring Low-Code Development: A Comprehensive Literature Review, Complex Systems Informatics and Modeling Quarterly 36, 68–86 (2023). doi: 10.7250/csimq.2023-36.04.

[13] F. Khorram, J.-M. Mottu, G. Sunyé, Challenges & Opportunities in Low-Code Testing, in: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS ’20. New York, NY, USA: Association for Computing Machinery (2020). doi: 10.1145/3417990.3420204.

[14] I. H. Azmy, A. Azmi, N. Kama, H. M. Rusli, S. Chuprat, A. W. Anuar, Methods for Application Development by Non-Programmers: A Systematic Literature Review, in: Proceedings of the 2023 5th World Symposium on Software Engineering, WSSE ’23, pp. 1–8. New York, NY, USA: Association for Computing Machinery (2023). doi: 10.1145/3631991.3631992.

[15] T. Cerny, M. J. Donahoo, M. Trnka, Contextual understanding of microservice architecture, ACM SIGAPP Applied Computing Review 17(4), 29–45 (2018). doi: 10.1145/3183628.3183631.

[16] A. Bellemare, Building Event-Driven Microservices, O’Reilly Media, 2020.

[17] T. C. Lethbridge, Low-Code Is Often High-Code, So We Must Design Low-Code Platforms to Enable Proper Software Engineering. In: Margaria, T., Steffen, B. (eds.), Leveraging Applications of Formal Methods, Verification and Validation, pp. 202–212. Springer International Publishing (2021).

[18] M. Weske, Business Decision Modelling, in: Business Process Management, pp. 241–257. Springer, Berlin, Heidelberg (2019). doi: 10.1007/978-3-662-59432-2_5.

[19] P. McFedries, Web Coding & Development All-in-One for Dummies. Wiley, 2018.

[20] L. Rockoff, The Language of SQL. Pearson Education, 2021.