Automated Synthesis Method of "Smart" Home Systems Based on the Architectural Pattern Redux

Vasyl Teslyuk [1][0000-0002-5974-9310], Artem Kazarian[1] [0000-0002-6883-0233],

Natalia Kryvinska [2][0000-0003-3678-9229], Ivan Tsmots[1][0000-0002-4033-8618],

Taras Teslyuk[1][0000-0001-6585-3715]

[1] Lviv Polytechnic National University, Lviv 79013, Ukraine
vasyl.m.teslyuk@lpnu.ua
[2]Comenius University in Bratislava, Odbojárov 10, Bratislava, Slovak Republic
Natalia.Kryvinska@fm.uniba.sk

Abstract. A method for design of smart home systems using the Redux architectural pattern has been developed. The method is based on the adaptation of the Redux architectural pattern of visual interfaces design for usage in the Internet of Things. Based on the developed method, a system of "smart" home for control of lighting devices with the help of motion sensors and lighting in the premises of office building was built. The developed design method allows to increase reliability indicators and to increase system performance. Improved reliability is achieved by reducing the number of direct relationships between system components. Also, the developed design method helps to reduce the amount of information that is duplicated in different components of the system by using one common data store to save the state, which increases the speed of updating the states of the system and the speed of lighting fixtures settings changes. The advantages of using the developed design method are experimentally demonstrated by emulating the work of the "smart" home system with the subsequent storage and analysis of lighting settings change speed before using and after applying the architectural Redux pattern.

Keywords: design; architectural pattern; Redux; "Smart" home.

1 Introduction

Every year, technical systems become more complex and start to use intelligent methods and models in their logic [1, 2, 3], which allows to significantly improve the initial parameters of the developed devices. The usage of approaches of technical systems intellectualization provides improvements in such parameters as energy consumption [4], expansion of system functionality, reliability, etc. Today, we are seeing the rapid development of smart home (SH) [5-7] and "smart" city technologies [2, 8, 9] that allow a high level of comfort for residents, reduced energy costs and improved management efficiency system. The introduction of intelligent technologies in indus-

2019 DCSMart Workshop.

trial production - enables to increase its energy efficiency [10, 11], environmental friendliness, reduce production costs, etc. In particular, technologies of "smart" production are actively developing. We observe large-scale implementations of intellectual technologies in military sphere, in the field of augmented reality technologies [12-14], medicine [15, 16], education [17] and others.

Thus, the requirements for smart home systems and other intelligent systems are becoming higher every year, while IoT devices developers are forced to manage an increasing number of states at certain times of system operation [18]. These states may include data from sensors located inside the house, stored historical sensor data, as well as information generated during system operation, such as commands for changing the operating modes of household appliances in Smart Homes.

Implementation of the system state management logic with often changes is an integral part of the smart home systems development process that requires considerable time and financial cost. If the state of the sensor requires updating the of a household appliance work mode, this situation can change the value of the state of another sensor that requires updating the work settings of a different household appliance in the home. At some stage of the SH software development, developers no longer know what's going on in the middle of the system's logic and can no longer control when, why, and how system states are updated. This situation, when the system becomes complicated, carries huge risks in terms of reliability of its operation [19, 20], speed of detection and correction of errors of the developed logic and possibility to add new functionality. Such complexity arises from the fact that two different concepts are inherently mixed: change and asynchrony. Changes mean the flow of data generated from real-time sensors located in the home, which affect changes in the settings of household appliances in the home. Asynchronous refers to the non-periodicity of these changes occurrence in indicators, such as changes of the sensor parameters that occur after the events in the house (human movement, temperature changes in the room), which do not depend on time intervals and do not have a clear timetable. To solve this problem, it is proposed to use the architectural pattern Redux with the adaptation of its concepts to the field of "smart" home systems development [21, 22].

The problem of system performance, which operates with many simultaneously generated events, is described in [23, 24, 25], where the authors provide examples and comparisons in usage of monolithic, microservice and multilevel architectures. In the field of visual user interface processing, the problem of large arrays of events processing is solved by the usage of Flux-like architectures, such as Redux, which implementation is also discussed in [26, 27], but the concepts and examples discussed are limited by the usage for the design of visual interfaces only.

It is clear that each of the above approaches [28, 29, 30] has its advantages and disadvantages and depends on the specific technical solution of the large-scaled "smart" home system, which is characterized by a system of requirements, such as: scalability, speed and reliability. The presented work is a further development of a research [31] related to development of method for designing SH systems using a Redux pattern for controlling home appliances. From the results of the analysis it follows that the best performance of the system was obtained during the implementation of Flux architecture, namely the architectural pattern Redux. Most slowly, the system worked when

using a monolithic architecture. Therefore, the analysis of the application effectiveness research results implemented with different approaches showed the feasibility of using the Flux architecture in the process of large-scaled "smart" home systems development.

2 The method of the architectural pattern Redux usage in design of "smart" home systems

Consequently, smart home systems are evolving every year. On the one hand, the level of intellectualization increases, and on the other hand - it increases the scale of such systems (number of components: sensors, actuators, microcontrollers, etc.). The development and analysis of such systems is impossible without modern design technologies.

The above analysis shows the feasibility of using the Flux pattern architecture in the process of SH systems synthesis. Redux is an architectural pattern for data state management, initiated in the field of web application development [32]. It is suitable for systems where state management can become complicated and confusing over time. Redux is not associated with a particular framework, although it was developed for the React Visual Interface Development Library. Redux proposes to keep the general state of the system parameters in the single storage. System components (sensors and household appliances) "send" state changes to system centralized storage rather than directly to other components. This solution helps to prevent confusion with data sources and data incorrect sending due to the increased complexity of internal logic or data flow rules in the system in case of adding new system functionality. The components that need to be in touch with these changes are "subscribed" to the storage. The storage can be considered as an "intermediary" for all changes in the parameters state of the devices and sensors in the house. With Redux the components do not communicate directly with each other; all changes must go through a single source - storage. With Redux, all components get their states from storage. It is also simple and clear where the component should send the status change information - to a single storage location. Each individual system component only initiates the change and does not care about the states of the other system components that should receive the change.

The block diagram of the built-in algorithm for the synthesis of the "smart" home system using Flux architecture is developed, shown below and provides for the following steps:

Step 1. Enter the input data.

Step 2. Analysis of baseline data and identification of technical solution problems. If there is a problem of system performance (high values of time delays), then go to step 3. Otherwise, shut down the algorithm.

Step 3. Generate options for system architectural solutions to solve the problem. If Yes, Go to Step 8.

Step 4 Research the current architectural solution.

Step 5. Implement the current architectural solution.

Step 6. Define and save the performance parameters of the system. Go to Step 4.

Step 7. Comparison of the obtained results of performance of the projected system.

Step 8. Choosing an Effective Architectural Solution.

Step 9. Implementation of the architectural solution. Shut down the algorithm. Implemented approach of the large-scaled "smart" home systems synthesis, using the architectural pattern of the Redux data stream, allows to increase the efficiency of the project solution implementation.

3 Adaptation of Redux architectural design concepts to the development of smart home systems

The basic concepts of the Redux architectural pattern include events, storage, and reducer. Events will mean structures that transmit data to the repository. They are the only sources of information for the data warehouse. In the field of SH development, an event is understood as a data structure created by an event generator and is a result of situations such as triggering of the encoder or the system user command. The event has a mandatory string parameter - a unique event name that identifies a specific unit of data structure in the stream. An example of such a parameter would be "Motion sensor trigger" when system fixed the motion in the living room. The event may also have a second parameter that contains additional useful information about the event, such as the time of motion sensor triggered, etc. An example of the event data structure is shown in Table 1.

Table 1. Example of event data structure.

Event name	Event information
«Motion sensor trigger»	room name: "Living room", room ID: 003, dateObjection: “19-03-2019”, timeObjection: “17:03:15”

Generally, the storage is an object that stores system state, provides access to system status, has the ability to update system state, and allows to register as a listener to update system state to receive new system state settings after changes have been made. Redux uses only one data store for the entire state of the system settings. Because the state of the system is located in a single place, it is called the single source of truth. The structure of the data storage is entirely dependent on the developers of the smart home system, but for real application it is usually an object with several levels of nesting. The system state repository is stored in the database of the developed system and does not depend on the model of the database used (relational / non-relational) and the database distribution type (centralized/distributed). The main requirement for the storage is the provision of a standardized interface that will allow all or some of the storage parameters to be obtained, to update the state in accordance with events that occur, and to subscribe to the repository state updates.

The internal structure of the repository is divided into the separate parts, which are called groups. Groups differentiate the data stored in the repository according to different data types and logical groups. In smart home systems, data is divided into groups based on belonging to specific individual rooms of the system (all indicators of sensors and modes of household appliances in a particular room), or the type of stored data, such as temperature values, data about the presence of residents, data about current modes of household appliances and more.

The data storage of the developed smart home system store the data grouped according to the rooms in the house. For example, let's consider a storage designed for a three-room house, where there are 5 groups in the data storage, related to the rooms with kitchen and bathroom: living room, bedroom, study, kitchen and bathroom.

The storage group is separated from other groups according to a parameters state related to current room. Each group contains the following data set: room temperature; humidity; presence of residents in the room (determined by the work of the motion sensor); room lighting; list of household appliances present in the room; customization of specific household appliances.

The internal structure of the data storage is divided into groups and is shown in Figure 1.

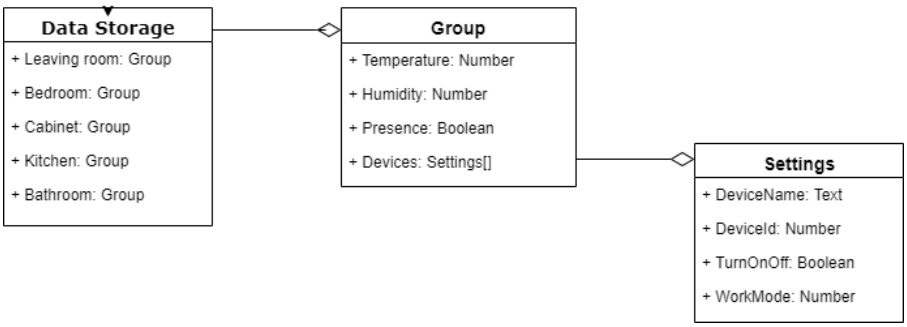

Fig. 1. The structure of the developed data storage.

Redux does not allow system components to change state directly. The events describe what changes should be made to the system settings state. Event handlers who can modify the storage are called reducers. The input parameters must necessarily get the current state of the system as well as the event data. According to the type of event generated with the unique name, the reducer performs logical actions on the parameters of the received state and returns the changed state of the system to the output. In case of receiving arguments of a certain type, the reducer must calculate the new version of the state and pass it to the data storage. There are no changes to the current state of the system, which provides a clear separation of the reducer functionality and the data storage. The work of the reducer is limited only by the computation of a new version of the state, which overwrites the current state of the system in the storage. The scheme of the system state life cycle is shown in Figure 2.

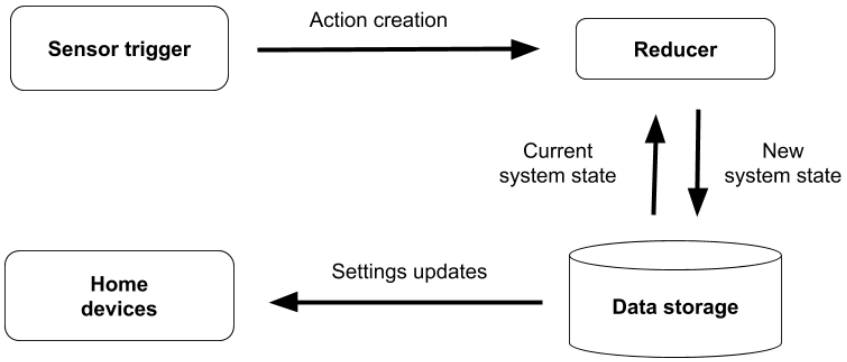

Fig. 2. The life cycle diagram of the "smart" home system state.

When the event is generated, the reducer alternately compares the type of received event with the known event types. In case of matching types, the reducer gets the current state of the system and using the system state change logic, changes the state parameters according to a specific type of event. The output result is a new state of the system stored in the data storage replacing the previous one. An example of the reducer work algorithm, which logic is responsible for maintaining the temperature mode in accordance with the data obtained from the temperature sensor and by changing the settings of the air conditioner or heater is given below.

Step 1. Enter the input data.

Step 2: Read the temperature sensor event ('event').

Step 3. Read the current system state ('state').

Step 4. If the air temperature in the room is higher than 21ºC, then switch on the air conditioner. Go to Step 8.

Step 5. If the room temperature is higher than 18ºC and less than 21ºC, switch off the heater and the air conditioner. Go to Step 8.

Step 6. If the room temperature is below 18ºC, switch on the heater and go to Step 8.

Step 7. If a temperature reading error is received or the value obtained is not a number, then return the current state of the system unchanged ('state').

Step 8. Save the new system status to the database. End the algorithm work.

Therefore, the smart home system developed is based on an adaptation of the Redux architectural pattern concepts, allow efficient control of temperature modes in the rooms and allow to scale the designed system with a larger number of rooms, without loss of performance due to the usage of a single storage approach and standard system storage data change flow regardless of the events source that occur in the home.

4 Advantages of using Redux in smart home systems

The advantages of the chosen architectural solution usage are clearly understood on the example of SH systems with a large number of rooms and a large number of sensors and household appliances. The benefits of using Redux will be reflected by monitoring the performance of the smart home system designed for the office, which consists of two rooms, a waiting room, a corridor, a bathroom and a large work area divided into the separate work zones. The created system receives data from the motion and light sensors and controls lighting throughout the office using a controller based on a single Raspberry PI microcomputer [33, 34]. Sensors and lighting have the following distribution by rooms shown in Table 2.

Table 2. Distribution of "smart" home system sensors and devices by rooms

Room name	Motion sensors	Light sensors	Lighting appliances
Room 1	3	2	5
Room 2	2	2	4
Waiting room	2	1	2
Corridor	3	2	3
Work area. Zone 1	3	2	6
Work area. Zones 2	2	1	5
Bathroom	1	1	3

Initial implementation of the system used the approach of direct communication between the sensors located in the office. When an sensor is triggered by a movement of a person in a room, the motion sensor with the integrated controller sends a request to the light sensor to obtain information about the current value of the lighting in the room. After calculating the light value, the light sensor sends the result to the controller. Based on internal logic, if the indoor lighting is insufficient, the controller sends a command to turn on the indoor lighting and send updated room parameters state (recent motion detection, illumination and lighting mode) to all other controllers connected to the system for the possibility of restoring the previous parameters in case of the controller room with a temporary power outage during power outages.

While further extension of the office space and adding of new rooms to the system management, this approach has proven to be difficult for scaling and with a clear tendency to reduce system performance while increasing the number of new sensors and appliances added to the system. Scalability and further development of the system with the direct component (sensors, devices) communication approach used by the system is limited by the complexity of component relationships in terms of system states changes and notification of all other components about a status updates. With the direct interaction of motion sensors, lighting sensors and lighting appliances, the diagram of the interconnections between all sensors and devices in the developed system looks like shown in Fig. 3.

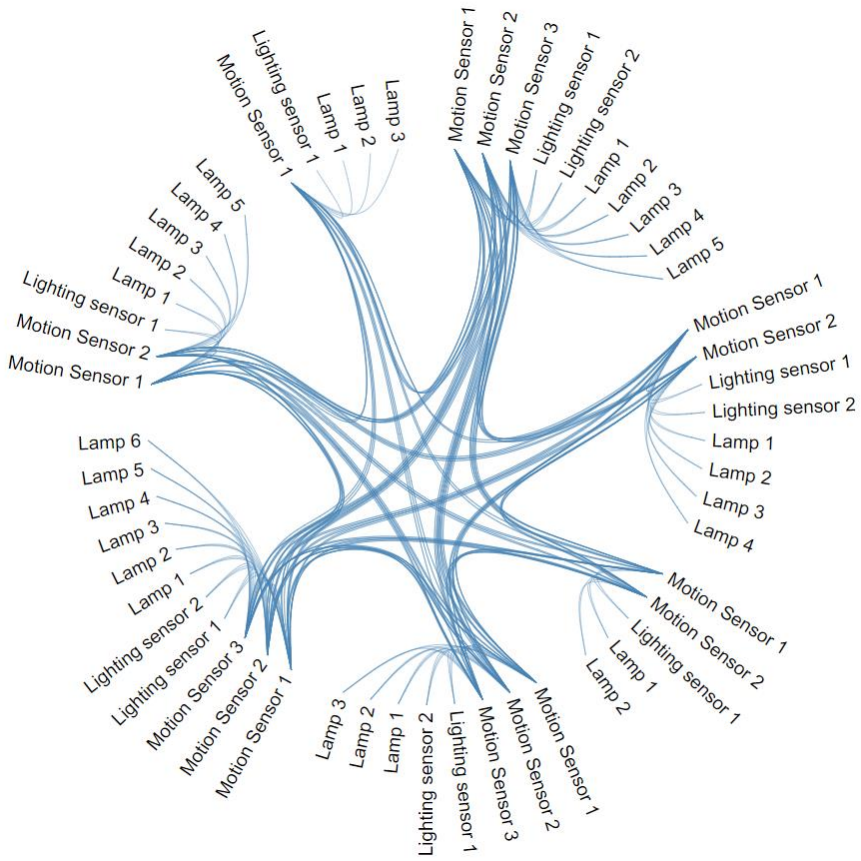

Fig. 3. Diagram of connections between sensors and devices of the SH system.

Performance analysis of the developed SH system was based on a randomly selected set of system state changes with stored timestamps at each step of receiving and sending commands between system components. The first sample contains a set of the 30 randomly selected records of state changes using the direct interaction between system components approach. The second sample contains a set of the 30 randomly selected records of state changes after the usage of Redux architectural pattern.

Using the Redux architectural pattern, the sensors and devices do not interact directly with each other. All sensors send events about their values changes to the reducer, which changes the state of the system in the data storage in accordance with the developed internal logic for responding to a specific event. The devices that subscribed to "listening" of changes in the data storage, change their own work settings when there is a change in the system state parameters. The relationships between all sensors and devices in a developed system using the Redux architectural pattern are shown in Figure 4.

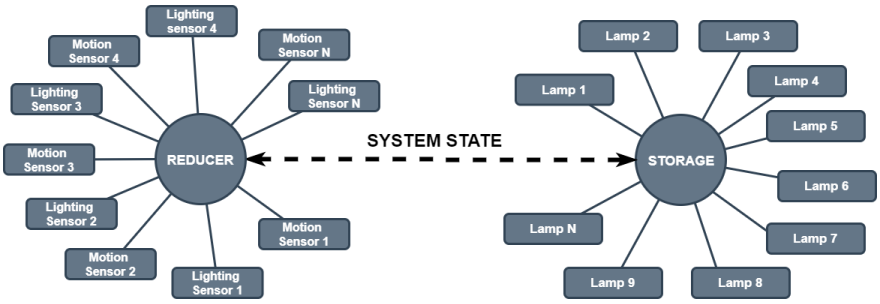

Fig. 4. Diagram of connections after the usage of Redux architectural pattern.

The analys of the results obtained and the comparison of parameters allow to see the advantages of using the architectural pattern Redux in the field of "smart" home systems. These advantages are the reduce of the connections number between different system components, which increases the system performance and improves overall reliability. Also, by reducing the number of connections between system components, accelerated the time of the system state change. The usage of Redux architectural pattern reduced the number of requests between system components from 120 requests to 60, which is a 50% decrease in comparison with initial value received before the system update. The speed of the general system state updates increased on 0.396 seconds, which is 46% of the system performance acceleration in comparison with the previous values of system speed before its update.

A comparison of the connections quantities between system components occurring in the developed system when the system components interact directly with each other and after the usage of Redux architectural pattern is shown in Figure 5.

Comparison of the general system state updating speed with the direct interaction of system components with each other and after the usage of Redux architectural pattern is shown in Figure 6.

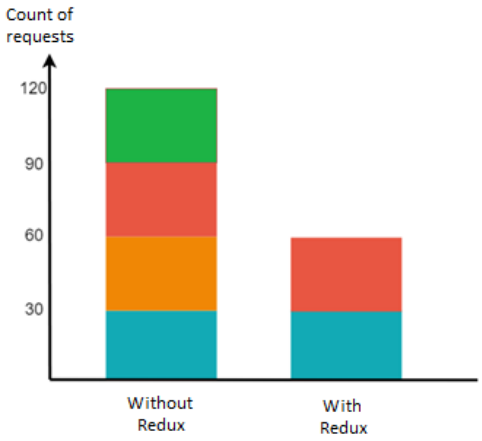

Fig. 5. Comparison results of connections quantities between components of developed system

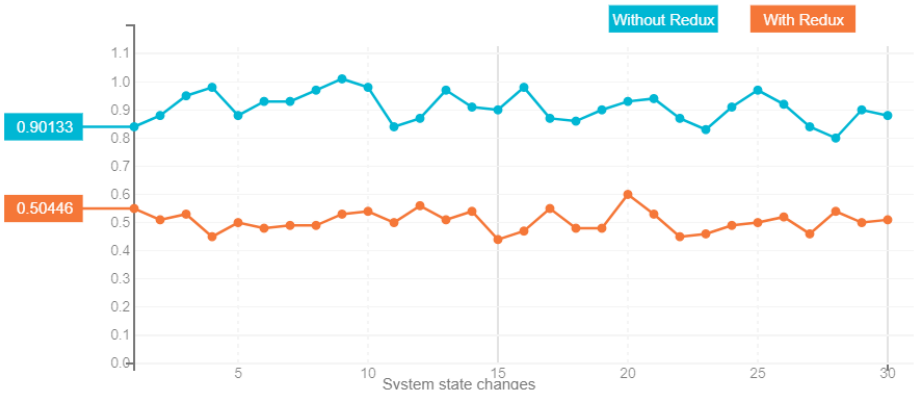

Fig. 6. Comparison of the general system state updating speed

The results show that the usage of the Redux architectural pattern reduces the number of connections between system components on 50%, which allows to increase the general system state updating speed and system performance on 46%.

5 Conclusions

The developed architecture of the smart home based on the Redux pattern provides a centralized mechanism for managing of the system data states in case of work with large arrays of events that occur during the work of smart home.

A method has been developed based on the usage of a Redux architectural pattern with a centralized data storage, which can significantly reduce the number of connections between system components, which increases its reliability, as well as improves performance of the system.

The algorithm of the method realization is implemented and researched the advantages of using the developed method algorithm for "smart" home systems using the architectural pattern Redux by the comparison of system performance parameters before and after the implementation of the proposed solution.

References

1. Poniszewska-Maranda, A., Kaczmarek, D., Kryvinska, N. et al.: Studying usability of AI in the IoT systems/paradigm through embedding NN techniques into mobile smart service system. Computing, 1–25 (2018).
2. Boreiko, O., Teslyuk, V., Zelinskyy, A., Berezsky, O.: Development of models and means of the server part of the system for passenger traffic registration of public transport in the "smart" city. Eastern-European Journal of Enterprise Technologies 1,2 (85), 40–47 (2017).
3. Cai, H., Xu, B., Jiang, L., Vasilakos, A.V.: IoT-Based Big Data Storage Systems in Cloud Computing: Perspectives and Challenge. IEEE Internet of Things Journal 4(1), 75–87 (2017).

4. Teslyuk, T., Tsmots, I., Teslyuk, V., Medykovskyy, M., Opotyak, Y.: Architecture and Models for System-Level Computer-Aided Design of the Management System of Energy Efficiency of Technological Processes at the Enterprise. In: Shakhovska N., Stepashko V. (eds) Advances in Intelligent Systems and Computing II. CSIT 2017. Advances in Intelligent Systems and Computing 689, Springer, Cham.: 538–557 (2018).
5. Sultan, M., Ahmed, K. N.: SLASH: Self-learning and adaptive smart home framework by integrating IoT with big data analytics. Computing Conference pp. 530–538, London (2017).
6. Kazarian, A., Teslyuk, V., Tykhan, M., Mashevska, M.: Usage Of SaaS Software Delivery Model In Intelligent House System. Przegląd Elektrotechniczny 95(7/2019), 38–41 (2019).
7. Biljana L. Risteska Stojkoska, Kire V. Trivodaliev: A review of the Internet of Things for smart home: Challenges and solutions. Journal of Cleaner Production 140 (3), 1454–1464 (2017).
8. Wilson, P.: State of smart cities in UK and beyond. IET Smart Cities 1(1), 19–22 (2019).
9. Tai-hoon, K., Carlos, R., Sabah, M.: Smart City and IoT. Future Generation Computer Systems 76, 159–162 (2017).
10. Jeehyeong, K., Guejong, J., Jongpil J.: A Novel CPPS Architecture Integrated with Centralized OPC UA server for 5G-based Smart Manufacturing. Procedia Computer Science 155, 113–120 (2019).
11. Carvalho, A., O'Mahony, N., Krpalkova, L., Campbell, S., Walsh, J., Doody, P.,: At the Edge of Industry 4.0. Procedia Computer Science 155, 276–281 (2019).
12. Petukhov, I., L. Steshina, L., Glazyrin, A.: Application of virtual reality technologies in training of man-machine system operators. International Conference on Information Science and Communications Technologies (ICISCT), pp. 1–7, Tashkent (2017).
13. Gorokhova, R.: Application Features Of Virtual Reality In Diagnostics Of Human Psychophysiological Characteristics. Conference: RPTSS 2017 International Conference on Research Paradigms Transformation in Social Sciences. pp. 411-419 (2018).
14. Martin, J., Dikkers, S., Squire, K. et al.: Participatory Scaling Through Augmented Reality Learning Through Local Games. TechTrends 58(1), 35–41 (2014).
15. Tkachenko, R., Izonin, I., Vitynskyi, P., Lotoshynska, N., Pavlyuk, O.: Development of the Non-Iterative Supervised Learning Predictor Based on the Ito Decomposition and SGTM Neural-Like Structure for Managing Medical Insurance Costs. Data 3(4), 1–14 (2018).
16. Berezsky, O., Verbovyy, S., Pitsun, O.: Hybrid Intelligent information techology for biomedical image processing. In: Proceedings of the IEEE International Conference of Computer Science and Information Technologies, pp. 420–423, Lviv (2018).
17. Hvorecký, J., Dávideková, M.: Social Life in Virtual Universities. In book: Teaching and Learning in a Digital World. Advances in Intelligent Systems and Computing 715, 78–85 (2018).
18. Teslyuk, V., Denysyuk, P., Al Shawabkeh H. A. Y., Kernytskyy, A.: Developing the information model of the reachability graph.In: Proc. of the 15-th International Seminar/Workshop on Direct and Inverse Problems of Electromagnetic and Acoustic Wave Theory, DIPED'2010, pp.210–214, Tbilisi, Sept. 27-30 (2010).
19. Engineering of Event-Based Systems. In: Distributed Event-Based Systems. Springer, Berlin, Heidelberg, pp.129–148.
20. Bobalo, Yu., Seniv, M., Yakovyna, V., Symets, I.: Method of Reliability Block Diagram Visualization and Automated Construction of Technical System Operability Condition. Advances in Intelligent Systems and Computing III, 871: 599–610 (2019).

21. Islam, Naim N. ReactJS: An Open Source JavaScript Library for Front-end Developement [Available by URL]. Metropolia University of Applied Sciences. – 2017. – [Electronic resource]. Access mode: https://www.theseus.fi/bitstream/handle/10024/130495/FInal_Year_Thesis.pdf?sequence=1&isAllowed=y
22. Piispanen, M. Modern architecture for large web applications (2017) [Electronic resource]. Access mode: https://jyx.jyu.fi/bitstream/handle/123456789/54129/1/URN%3ANBN%3Afi%3Ajyu-201705272524.pdf
23. Paul, A., Nalwaya, A.: Flux: Solving Problems Differently. In: React Native for iOS Development. Apress, Berkeley, CA, 75–93 (2016).
24. Freeman, A.: Using a Redux Data Store. In: Pro React 16. Apress, Berkeley, CA, pp.531–559 (2019).
25. Saransig, A., Tapia, F.: Performance Analysis of Monolithic and Micro Service Architectures – Containers Technology. In: Mejia J., Muñoz M., Rocha Á., Peña A., Pérez-Cisneros M. (eds) Trends and Applications in Software Engineering. CIMPS 2018. Advances in Intelligent Systems and Computing, vol 865, pp.270–279, Springer, Cham (2019).
26. Familiar, B.: From Monolithic to Microservice. In: Microservices, IoT, and Azure. Apress, Berkeley, CA, 1–7 (2015).
27. Nene, A.V., Joseph, C.T., Chandrasekaran, K.: Construing Microservice Architectures: State-of-the-Art Algorithms and Research Issues. In: Uden L., Ting IH., Corchado J. (eds) Knowledge Management in Organizations. KMO 2019. Communications in Computer and Information Science, vol 1027, pp. 364–376, Springer, Cham (2019).
28. Christudas, B.: Microservices in Depth. In: Practical Microservices Architectural Patterns. Apress, Berkeley, CA, 35–53 (2019).
29. Kalske, M., Mäkitalo, N., Mikkonen, T.: Challenges When Moving from Monolith to Microservice Architecture. In: Garrigós I., Wimmer M. (eds) Current Trends in Web Engineering. ICWE 2017. LNCS, vol 10544, pp. 32–47, Springer, Cham (2018).
30. Gackenheimer, C.: Introducing Flux: An Application Architecture for React. In: Introduction to React. Apress, Berkeley, CA, pp. 87–106 (2015).
31. Kazarian, A., Teslyuk, V., Tsmots, I., Greguš, J.: Development of a «smart» home system based on the modular structure and architectural data flow pattern Redux. Procedia Computer Science 155, 35–42 (2019).
32. Molnár, E., Molnár, R., Kryvinska, N., Greguš, M.: "Web Intelligence in practice", The Society of Service Science. Journal of Service Science Research 6 (1), 149–172 (2014).
33. Bakir, A.; Setting Up a Raspberry Pi and Using It As a HomeKit Bridge. In: Program the Internet of Things with Swift for iOS. Apress, Berkeley, CA, 235–266 (2018).
34. Jayakumar, A.J.K., Muthulakshmi, S.: Raspberry Pi-Based Surveillance System with IoT. In: Thalmann D., Subhashini N., Mohanaprasad K., Murugan M. (eds) Intelligent Embedded Systems. Lecture Notes in Electrical Engineering, vol 492, pp. 173–185, Springer, Singapore (2018).