Information systems. Computer sciences. Issues of information security

Информационные системы. Информатика. Проблемы информационной безопасности

UDC 004.41
https://doi.org/10.32362/2500-316X-2022-10-5-16-27

RESEARCH ARTICLE

Framework for experimental evaluation of software solutions in a virtual environment

Dmitry Ilin @

MIREA – Russian Technological University, Moscow, 119454 Russia
@ Corresponding author, e-mail: i@dmitryilin.com

Abstract
Objectives. Ready-made information technology solutions used when developing software have various characteristics depending on the objectives to be experimentally obtained. While the selection of appropriate technologies and software tools used in experimental software engineering can be time-consuming, experimental complexity can be reduced by providing the researcher with domain-specific tools. The aim of the study is to design and develop a domain-specific software framework for experimental evaluation of the characteristics of information technology solutions in a virtual environment.
Methods. To determine the required characteristics of the software framework, an analysis of software tools for conducting experimental studies to evaluate the characteristics of information technology solutions in a virtual environment was conducted. Methods of decomposition, structural design, and software development were applied to design and develop the framework.
Results. A software framework for conducting experimental research has been developed. The design results, key features of the framework and a description of the functionality are presented. The implementation of the framework comprises commands for managing virtual machines and commands for scaffolding. A technique for conducting experimental studies using the framework is proposed.
Conclusions. The developed domain-specific software framework addresses shortcomings of existing tools to reduce labor costs when conducting experiments to evaluate information technology solutions. The developed framework and proposed methodology allows the number of programming and markup languages required for setting up a software experiment to be reduced from 3 to 1.

Keywords: software framework, virtual machines, experimental evaluation of software solutions, resource utilization monitoring, scaffolding

• **Submitted:** 04.03.2022 • **Revised:** 13.07.2022 • **Accepted:** 05.09.2022

For citation: Ilin D. Framework for experimental evaluation of software solutions in a virtual environment. *Russ. Technol. J.* 2022;10(5):16–27. https://doi.org/10.32362/2500-316X-2022-10-5-16-27

Financial disclosure: The author has no a financial or property interest in any material or method mentioned.

The author declares no conflicts of interest.

НАУЧНАЯ СТАТЬЯ

Программный фреймворк для экспериментальной оценки характеристик информационно-технологических решений в виртуальной среде

Д.Ю. Ильин @

МИРЭА – Российский технологический университет, Москва, 119454 Россия
@ *Автор для переписки, e-mail: i@dmitryilin.com*

Резюме
Цель. При разработке программного обеспечения, как правило, применяются готовые информационно-технологические решения. Они обладают различными характеристиками, объективные данные о которых можно получить экспериментально. Постановка корректного и воспроизводимого эксперимента требует от исследователя применения целого ряда разрозненных технологий и программных инструментов, что делает задачу трудоемкой. Снизить трудоемкость постановки эксперимента возможно, предоставив исследователю предметно-ориентированный инструментарий. Цель работы – проектирование и разработка предметно-ориентированного программного фреймворка для экспериментальной оценки характеристик информационно-технологических решений в виртуальной среде.
Методы. Для определения требуемых характеристик программного фреймворка проведен анализ программных инструментов проведения экспериментальных исследований по оценке характеристик информационно-технологических решений в виртуальной среде. При проектировании и разработке фреймворка применены методы декомпозиции, структурного проектирования, разработки программного обеспечения.
Результаты. Спроектирован и разработан программный фреймворк для экспериментальной оценки характеристик информационно-технологических решений в виртуальной среде. Представлены результаты проектирования, ключевые особенности фреймворка и программные технологии, примененные для разработки. Приведено описание функциональных возможностей фреймворка. Реализация фреймворка содержит 12 команд для управления виртуальными машинами и 4 команды для скаффолдинга. Предложена методика проведения экспериментальных исследований с применением фреймворка.
Выводы. Проведенное исследование позволило идентифицировать недостатки применения существующего инструментария, разработать предметно-ориентированный программный фреймворк и предложить методику его использования, что может сократить трудозатраты при проведении экспериментов по оценке информационно-технологических решений в виртуальной среде. Фреймворк позволяет сократить количество языков программирования и разметки, необходимых исследователю для постановки эксперимента, с 3 до 1.

Ключевые слова: программный фреймворк, виртуальные машины, экспериментальная оценка характеристик программного обеспечения, мониторинг вычислительных ресурсов, скаффолдинг

• **Поступила:** 04.03.2022 • **Доработана:** 13.07.2022 • **Принята к опубликованию:** 05.09.2022

Для цитирования: Ильин Д.Ю. Программный фреймворк для экспериментальной оценки характеристик информационно-технологических решений в виртуальной среде. *Russ. Technol. J.* 2022;10(5):16–27. https://doi.org/10.32362/2500-316X-2022-10-5-16-27

Прозрачность финансовой деятельности: Автор не имеет финансовой заинтересованности в представленных материалах или методах.

Автор заявляет об отсутствии конфликта интересов.

INTRODUCTION

Software development is largely based on the integration of ready-made information technology solutions including software libraries, frameworks, database management systems or entire software products.

Integrated solutions may have different quality characteristics, including those evaluated under different operating conditions [1–3]. The assessment of quality characteristics such as performance [2–4] is especially important when designing systems that serve a large number of users. To do this, the considered information technology solutions or the result of their integration are tested experimentally [3–5]. As a rule, such experiments are carried out on a cloud infrastructure, although this is not always advisable. In some cases, the necessary infrastructure can be organized at the workstations [6, 7] of researchers. In both cases, the organization of the infrastructure requires the preparation of virtual machines or containers [8, 9], which allow information technology solutions to be evaluated in an isolated environment.

Researchers generally agree that the results of experimental assessments should be reproducible [7, 10–13]. However, there are different approaches for ensuring reproducibility. In some works, the importance of detailed infrastructure documentation and experimental protocols [12] is noted; in others, tools are proposed for recording software actions at the level of system calls [10, 11]. In some cases, infrastructure is not manually prepared but with organized using widely used DevOps[1] technologies [14–17]. These technologies have proven themselves, including for research purposes [5, 18], as a means of ensuring the reproducibility of experiments. In addition, they can be used as part of pedagogical activities [19–21] to provide participants in the educational process with an identical working environment.

However, according to some studies, the implementation of a reproducible experiment requires the use of many software tools [10, 11], which requires additional knowledge and skills from the researcher. It is also noted that the use of such tools can increase labor costs when setting up experiments.

The present work is devoted to the analysis of tools used to implement infrastructure on researchers' workstations and the development of a framework for conducting experimental studies to evaluate the characteristics of information technology solutions in a virtual environment.

To achieve this goal, it is necessary to:

- analyze the tasks for the researcher when setting up the experiment, as well as to determine the tools used to solve them;
- based on the analysis, prepare architectural solutions that define the key characteristics of the software framework;
- develop a framework corresponding to the subject area and architecture solutions, as well as present a methodology for its application.

TOOLS FOR CARRYING OUT EXPERIMENTS FOR ASSESSING INFORMATION TECHNOLOGY SOLUTIONS

Before proceeding to the development of a software framework, it is necessary to analyze the tasks for the researcher when setting up a reproducible experiment, as well as the tools used to solve them.

Although experimental studies for assessing the characteristics of information technology solutions can be conducted without using the software tools listed below, the feasibility of their use is justified.

The use of virtual machines (for example, using *VirtualBox*[2] [9, 22]) makes it possible to conduct experiments using a smaller number of devices including conducting an experiment on a single physical device.

However, the virtual environment itself can take up a significant amount of disk space, making it difficult to transfer virtual machines between researchers. Moreover, the virtual environment does not provide tools for formally describing its own configuration or that of any virtual machines used. Thus, in post-factum descriptions of the experimental bench, there is a possibility that documenting errors will arise.

This can be compensated by using special tools for creating and configuring a virtual environment (for example, *Vagrant*[3] [9, 22]). If the virtual machines are defined using the configuration tool, a formal description of the characteristics of the virtual bench can be determined even prior to its operation. Cloud configuration tools (for example, *Ansible*[4] [22, 23]) can be used to manage the configuration of virtual machine software in a similar way. This "infrastructure as code" approach [14, 23] solves two identified problems: reducing the volume of the experimental bench when transferring between researchers and ensuring the reliability of experimental documentation.

[1] DevOps is an acronym for development & operations—a set of practices for automating the technological processes of building, configuring, and deploying software.

[2] https://www.virtualbox.org/. Accessed February 18, 2022.
[3] https://www.vagrantup.com/. Accessed February 18, 2022.
[4] https://www.ansible.com/. Accessed February 18, 2022.

However, such existing solutions to the described problems are not always sufficient for conducting experiments since the configuration of the virtual experimental bench does not always remain unchanged throughout the experiment. For example, restrictions on the utilization of computing resources may be required only at certain stages of the experiment. Since the above-described tools describe a static configuration, they cannot be used to change the characteristics of the bench.

A program-controlled experiment also requires a task manager (for example, *Gulp*[5]) that executes commands in a given order. Data collection tools are also important for monitoring the utilization of computing resources (for example, *Atop*[6]) and tools for preparing reports.

Summarizing the above, we can conclude that in order to conduct a well-documented program-controlled experiment in a virtual environment, it is necessary to prepare a virtual bench to solve the following tasks:

1) design of the experimental study;
2) preparation of a formal description of virtual machines;
3) preparation of a formal description of virtual machine software configuration;
4) installation of monitoring tools for resource utilization on each virtual machine (in addition to 3);
5) preparation of research materials (IT solution, experimental data, etc.);
6) development and debugging of the program code for managing the experiment;
7) ensuring correct restrictions on resource utilization at various stages of the experiment;
8) export of monitoring data and generation of a resource utilization report.

Fulfilling the above-mentioned conditions requires a researcher to know a large number of technologies and means of integrating them. The situation is also complicated by the fact that different technologies require knowledge of different programming languages and data markup. Table 1 shows a list of the technologies, as an example. It is important to note that knowledge of the listed technologies becomes mandatory.

Due to the significant list of tasks facing before the researcher, some of which are not solved by existing tools, it is necessary to design and develop a tool for:

- describing the experiment in the form of software configurations;
- using specific software configurations at specific stages of the experiment;
- reducing the number of programming languages, data markup and information technologies required for the experiment.

[5] https://gulpjs.com/. Accessed February 18, 2022.
[6] https://www.atoptool.nl/. Accessed February 18, 2022.

Table 1. Example of a list of technologies required for conducting experiments in a virtual environment

Category	Tool	Language (programming, markup)
Configuring virtual machines	*Vagrant*	Ruby[7]
Configuring Virtual Machine Software	*Ansible*	YAML[8]
Experiment management	*Gulp*, Shell script[9]	JavaScript[10], Bash[11]
Report generation	*Gulp* + *EJS*[12]+ *Plotly.js*[13]	JavaScript, HTML[14], JSON[15]
Export monitoring data	*Atop*, Shell script	Bash

FRAMEWORK DESIGN

In the process of designing the framework, an analysis of software requirements was carried out and a software architecture that meets the specified requirements was formed.

Since the configuration of a virtual machine and its software parameters are described statically, they can be specified using semi-structured formats. Such a format should be easy to read and edit, so it is suggested to use the YAML markup language. This language is used in the *Ansible* system for configuring virtual machine software and is familiar to researchers. Instead of using a general-purpose programming language, the algorithm for executing automated tasks can also be specified declaratively in YAML.

It is understood that the experimental framework will compile YAML source files into target configurations (for example, the configuration for the *Vagrant* tool in Ruby), and will contain a built-in task automation tool. Thus, the researcher, instead of using three different languages to configure virtual machines, their software and perform automated tasks, only needs the YAML markup language, as shown in Fig. 1.

In order to ensure that the experiment is carried out taking into account the configurations specific for each stage, the following is proposed. The experiment

[7] https://www.ruby-lang.org/. Accessed February 18, 2022 (in Russ.).
[8] https://yaml.org/. Accessed February 18, 2022.
[9] Shell script is a script written in Bash or a similar language.
[10] https://262.ecma-international.org/6.0/. Accessed February 18, 2022.
[11] http://www.gnu.org/software/bash/. Accessed February 18, 2022.
[12] https://ejs.co/. Accessed February 18, 2022.
[13] https://plotly.com/javascript/. Accessed February 18, 2022.
[14] https://html.spec.whatwg.org/multipage/. Accessed February 18, 2022.
[15] https://www.json.org/. Accessed February 18, 2022.

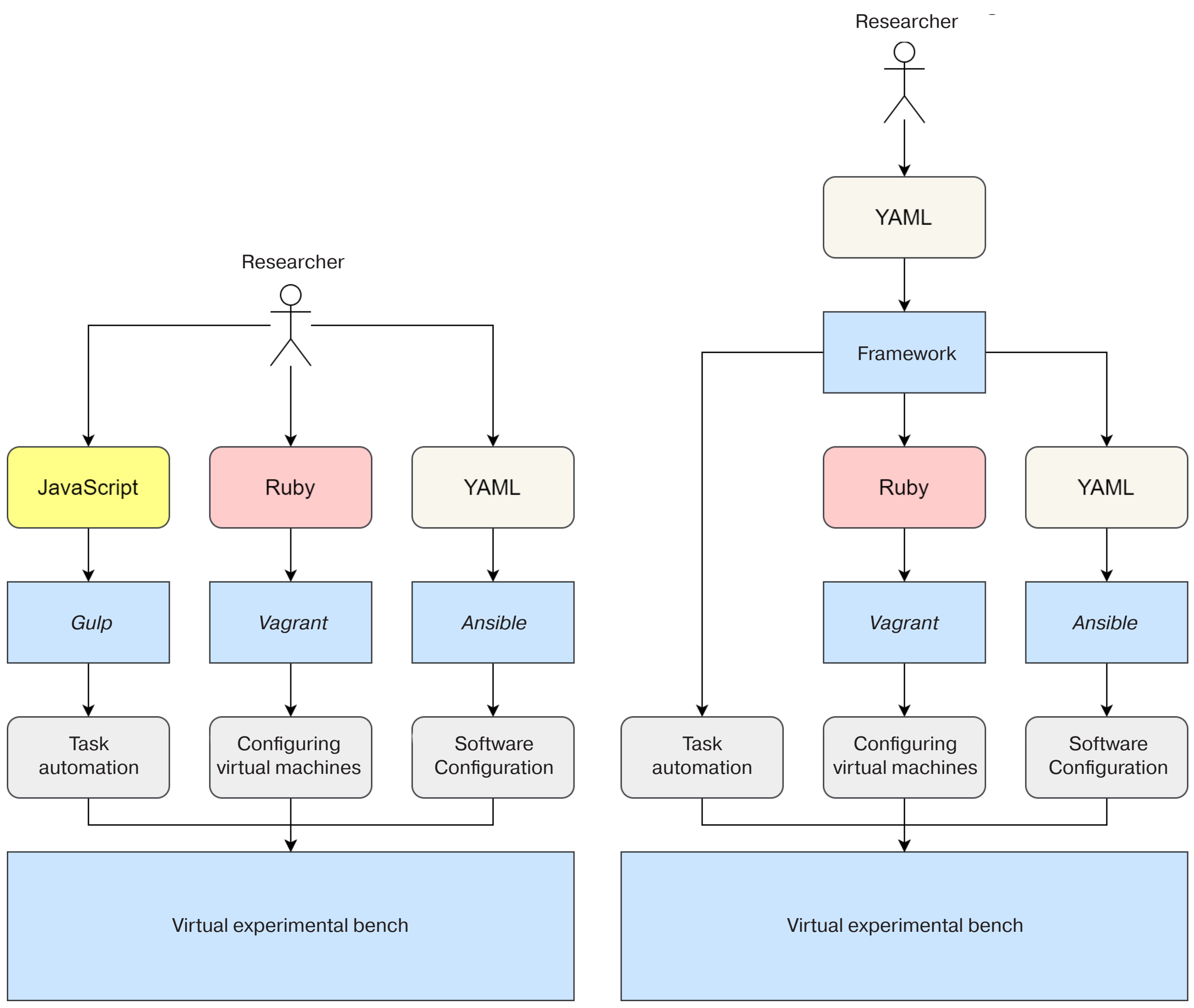

Fig. 1. Reducing the number of programming and data markup languages used by the researcher

is divided into stages, at each of which the compiled configurations are applied to each virtual machine. These consist of three components:

(1) a general configuration for the entire project;
(2) a configuration for a specific virtual machine;
(3) a configuration that addresses a specific stage of the experiment.

After compiling the selected configurations, a number of actions are performed to define the algorithm of the experiment at a particular stage. These actions can include any commands for virtual machines. The diagram illustrating the process of the experiment is shown in Fig. 2.

The file structure of the project for conducting an experimental study is shown in Fig. 3. A typical project consists of:

- configuration file;
- one or more virtual machines, for each of which there is also a separate configuration file and a set of provisioning files specific to the virtual machine;
- common provisioning files for all virtual machines;
- files with initial data (including the researched information technology solution);
- directories with reports on the utilization of virtual machine resources in the course of the experiments.

The experiment configuration file and virtual machine configuration files are implemented in a semi-structured YAML format. The structure of these files is schematically shown in Fig. 4, where at the top (file repex.yml) the configuration of the experiment is shown, and at the bottom (file vm.yml) the configuration of one of the virtual machines.

The content of the files is interconnected. The "path" property in the experimental configuration specifies the path to the vm.yml file. The experiment stage name (stage) used in the experiment configuration file can be used in the virtual machine configuration in order to apply its settings other than the default settings. These settings will be applied only at the specified stage of the experiment; however, the features of the hypervisor

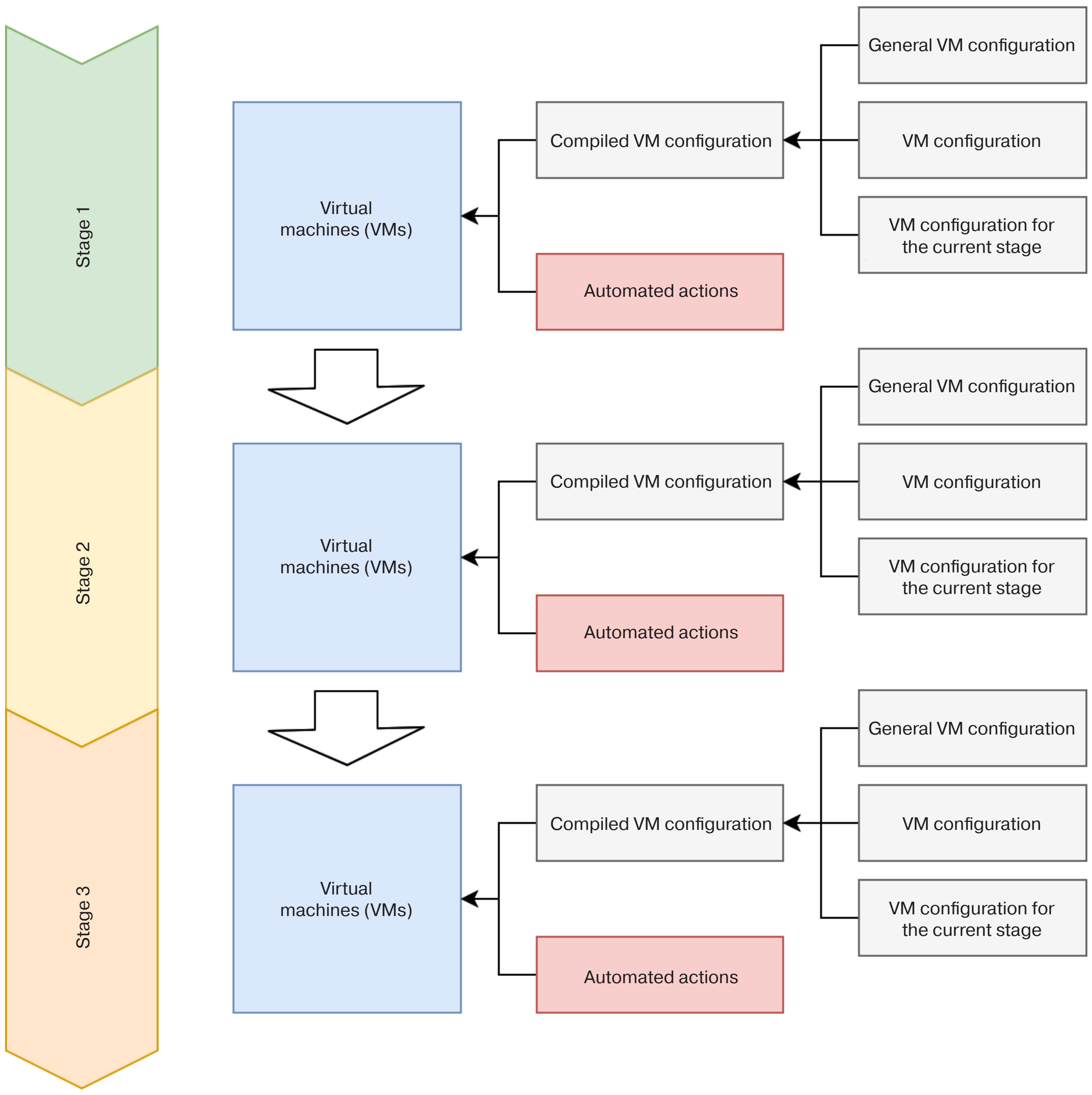

Fig. 2. Block-diagram of the experiment

should be taken into account. So, most of the settings will be applied only after rebooting the virtual machine. Some settings, such as disk subsystem or network bandwidth limits, are applied permanently, and you will need to create another stage of the experiment to remove the applied restrictions.

SOFTWARE IMPLEMENTATION OF THE FRAMEWORK

Based on the obtained design results, the *Repexlab* (reproducible experiment laboratory) software framework[16] was developed and published as an open source software. It is a tool with a command line interface (command line interface, hereinafter referred to as CLI), the main task of which is to support experimental studies in order to evaluate the characteristics of IT solutions. The framework sets the basic configuration of the project, ensures that the settings of the virtual environment are applied at various stages of the experiment.

The software framework is based on a number of existing tools:

VirtualBox is a virtualization system that allows you to organize the work of one or more virtual machines, including on a PC;

- *Vagrant* is a virtual environment creation and configuration tool. As a rule, it is used to prepare

[16] https://github.com/rnd-student-lab/repexlab. Accessed February 18, 2022.

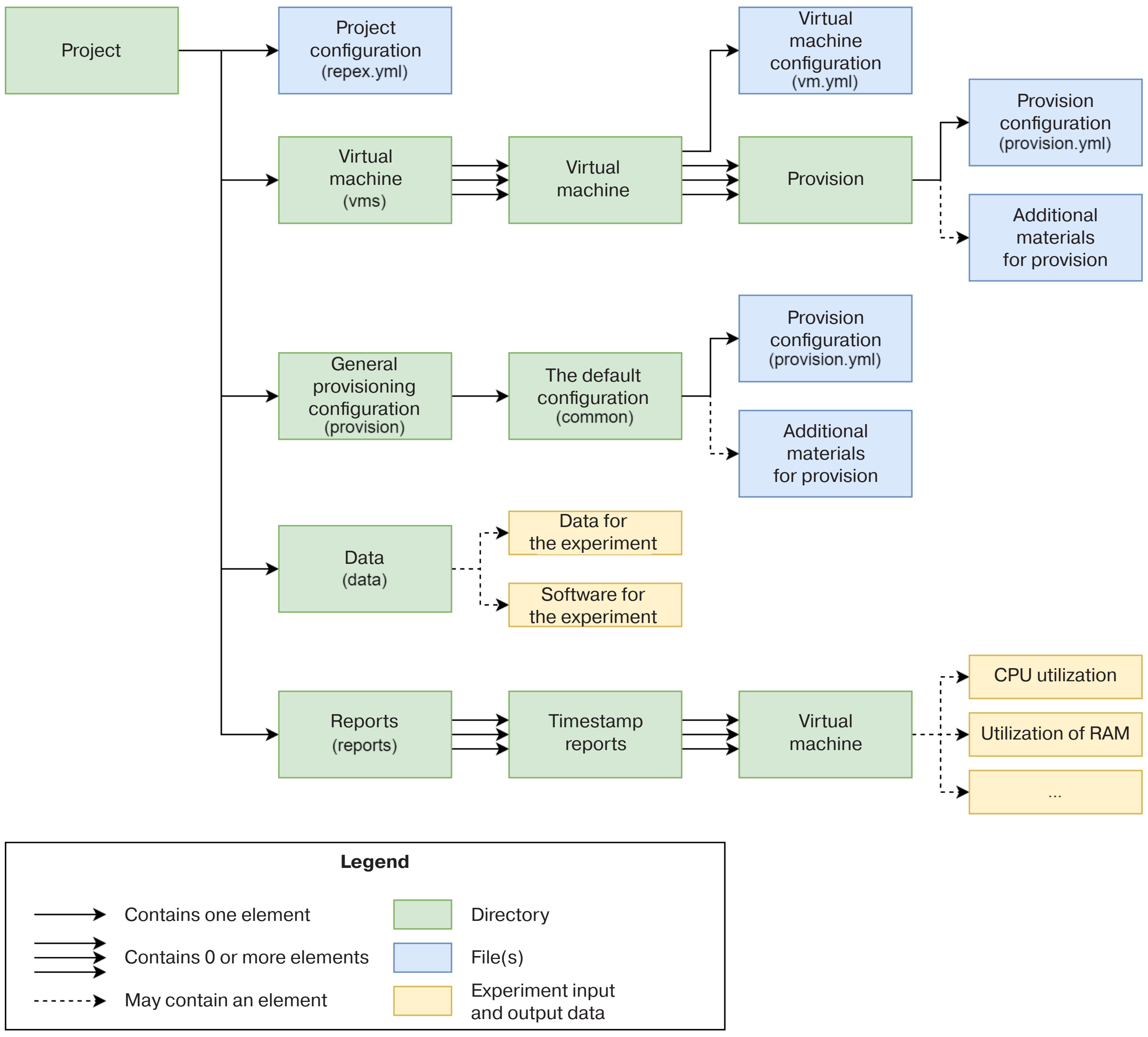
Project
Project configuration (repex.yml)
Virtual machine (vms)
Virtual machine
Virtual machine configuration (vm.yml)
Provision
Provision configuration (provision.yml)
Additional materials for provision
General provisioning configuration (provision)
The default configuration (common)
Provision configuration (provision.yml)
Additional materials for provision
Data (data)
Data for the experiment
Software for the experiment
Reports (reports)
Timestamp reports
Virtual machine
CPU utilization
Utilization of RAM
...
Legend
Contains one element
Contains 0 or more elements
May contain an element
Directory
File(s)
Experiment input and output data

Fig. 3. File structure of the project for the experimental study

a virtual working environment for developers. The base version uses *VirtualBox* as a hypervisor;

- *Ansible* is a cloud configuration management tool. Used by the framework as the main means of provisioning virtual machines;
- *Atop* is a tool for monitoring processes in the operating system. Used by the framework to collect data on the utilization of the computing resources of virtual machines;
- *Node.js*[17] and *NPM*[18] is a JavaScript code execution platform and an accompanying package manager. Since *Repexlab* is developed in JavaScript, they are required to install and execute the framework code.

The main functions that distinguish the framework from the totality of technologies used in it include:

- creation of a basic project for conducting an experimental study;
- scaffolding[19] for managing virtual machines within the experimental bench;
- control of the virtual bench from the command line;
- built-in tools for automating the tasks of interaction with a virtual bench.

The developed tool has commands that can be used both from the command line and with the help of built-in tools for automating tasks. They are divided into two categories: commands for interacting with virtual machines within an experiment (Table 2) and commands for changing the configuration of a virtual experimental bench (Table 3).

[17] https://nodejs.org/. Accessed February 18, 2022.
[18] https://www.npmjs.com/. Accessed February 18, 2022.

[19] Scaffolding is a programming method that involves generating program code for solving typical tasks, such as, for example, building a project file structure or creating classes for accessing database tables.

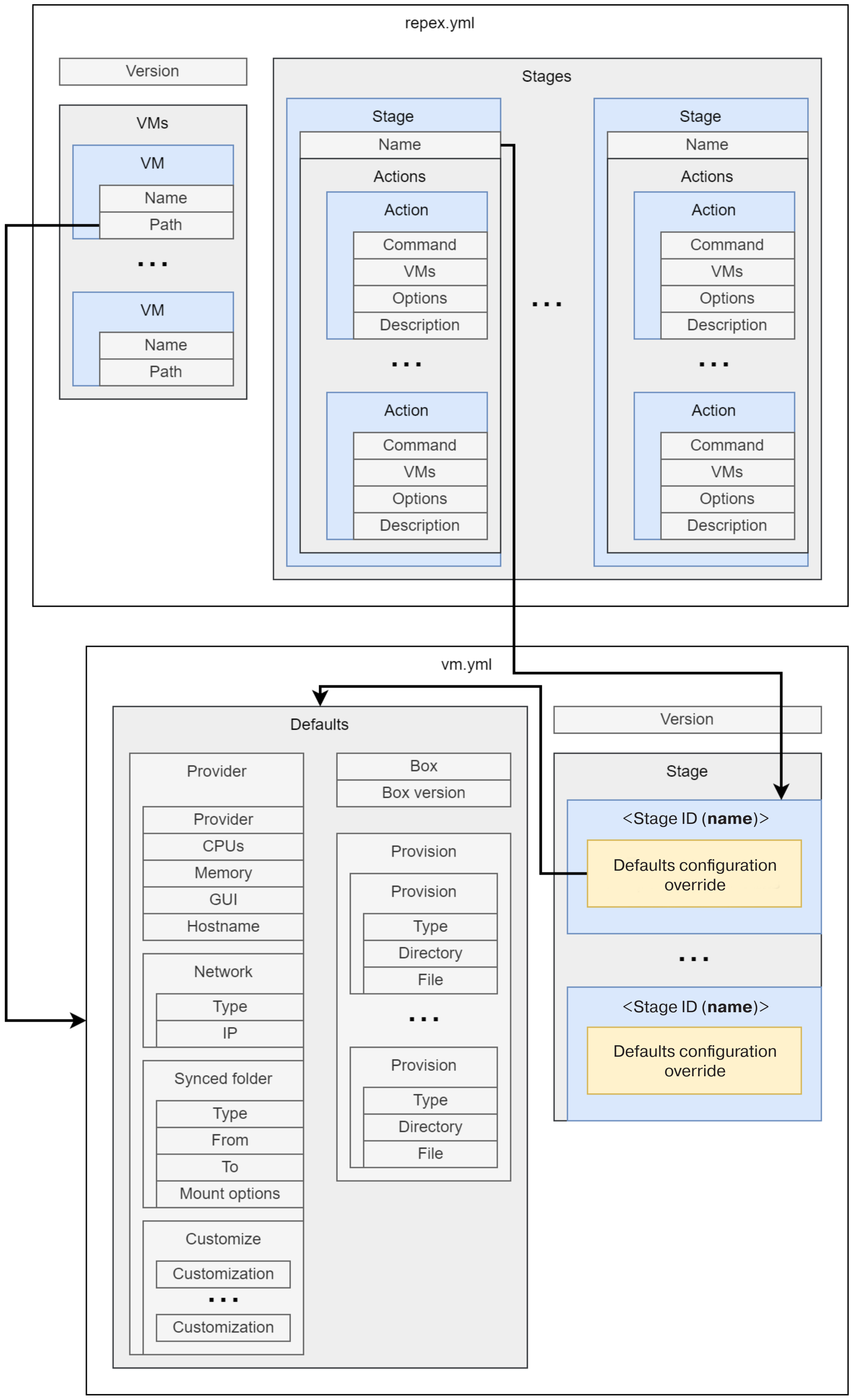

Fig. 4. Links to the configuration files of the project for conducting an experimental study

Table 2. List of commands for interacting with virtual machines

Command	Option		Command description
	Name	Description	
vm compile	-n, --name	Name of the virtual machine	Compiles virtual machine settings from a semi-structured data format to the configuration formats used by the tools
	-s, --stage	Experiment stage name	
vm copy	-n, --name	Name of the virtual machine	Copies a file or directory between the host system and the specified virtual machine
	-s, --stage	Experiment stage name	
	-d, --direction	Copy direction	
	-f, --from	Copy from	
	-t, --to	Copy to	
vm destroy	-n, --name	Name of the virtual machine	Deletes all data associated with the virtual machine (not configuration)
	-s, --stage	Experiment stage name	
vm exec	-n, --name	Name of the virtual machine	Execute a command on the specified virtual machine
	-s, --stage	Experiment stage name	
	-c, --command	Command to execute on a virtual machine	
vm provision	-n, --name	Name of the virtual machine	Performs installation and configuration of virtual machine software in accordance with the provisioning configurations
	-s, --stage	Experiment stage name	
vm report	-n, --name	Name of the virtual machine	Generates a report on the use of computing resources by virtual machines. In parentheses is the option in case of using automation
	-s, --stage	Experiment stage name	
	--start	The start time of the period for the report (or the name of the stage of the experiment)	
	--end	Report period end time (or experiment stage name)	
	-l, --labels	List of Atop labels to generate a report on them	
vm restart	-n, --name	Name of the virtual machine	Restarts the virtual machine
	-s, --stage	Experiment stage name	
vm setupHosts	-n, --name	Name of the virtual machine	Adds the IP-Hostname bindings of each virtual machine in the experiment to the /etc/hosts file of the virtual machines to simplify addressing
	-s, --stage	Experiment stage name	
vm ssh	-n, --name	Name of the virtual machine	Connecting to a virtual machine via SSH
vm start	-n, --name	Name of the virtual machine	Starts the virtual machine
	-s, --stage	Experiment stage name	
vm status	-n, --name	Name of the virtual machine	Displays the status of the virtual machine
	-s, --stage	Experiment stage name	
vm stop	-n, --name	Name of the virtual machine	Stops the virtual machine
	-s, --stage	Experiment stage name	

Table 3. List of commands for interacting with the project

Command	Option		Command description
	Name	Description	
project init	None, interactive parameter entry mode is used		Initializes the experiment project in the current directory and creates a base file structure
project run	-s, --stage	Experiment stage name	Starts the run of a configured experiment sequence or a specified experiment stage
project vm add	None, interactive parameter entry mode is used		Adds a new virtual machine configuration
project vm remove	-n, --name	Name of the virtual machine	Removes an existing virtual machine configuration

Since the commands from the Table 3 change the structure of the experiment, their use is possible only through the CLI. Most commands from the Table 2 are applicable both using the CLI for testing and debugging individual actions, as well as means of the built-in task automation tool. The exceptions are the "vm status" and "vm ssh" commands, which are only applicable via the CLI.

The application of the developed tool in practice is expected according to the following methodology:

1) planning and design of the experimental study;
2) initialization of the experimental study project with a given number of virtual machines;
3) preparation of a formal description of virtual machines and their configurations in YAML format;
4) development and debugging of the sequence of actions in the experiment using CLI tools;
5) automation of the sequence of actions in the experiment using tools for automating the execution of tasks;
6) conducting the experiment in a program-controlled mode;
7) analysis of exported data on the utilization of computing resources and other data obtained during the experiment.

CONCLUSIONS

In this work, tasks facing researchers when setting up experiments to assess the characteristics of information technology solutions have been analyzed along with the various tools used. It is shown that experimental studies require researchers who are knowledgeable and skillful at working with a large number of separate tools. Three key characteristics with which the framework must comply are formulated.

A domain-specific framework comprising some of the technologies used and providing the necessary functionality for conducting experimental research is presented. A scheme of interaction between the framework and the corresponding software for the experiment has been prepared. The general scheme of experiments based on virtual machines is determined. The formulated file structure of the experiment project includes the structure of the main project files and internal links between the project files.

A framework developed according to the results of the study contains 12 commands for working with virtual machines. Most commands can be executed both in CLI mode and in task automation mode. To simplify the process of preparing an experiment project, 4 scaffolding commands were implemented. The proposed methodology for using the framework in practice is described along with software technologies with which the framework is to be implemented.

Further research can be aimed at solving specific problems of evaluating information technology solutions using the framework, as well as identifying and eliminating the limitations of the framework, in order to develop the methodological basis for conducting experimental research using the proposed tools.

ACKNOWLEDGMENTS

The work was supported by the RTU MIREA grant "Innovations in the implementation of priority areas for the development of science and technology," Research Part project No. 28/22.

REFERENCES

1. Barrett E., Bolz-Tereick C.F., Killick R., Mount S., Tratt L. Virtual machine warmup blows hot and cold. In: *Proc. ACM Program. Lang.* 2017;1:52:1–52:27. https://doi.org/10.1145/3133876
2. Eismann S., Bezemer C.-P., Shang W., Okanović D., van Hoorn A. Microservices: A performance tester's dream or nightmare? In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, Edmonton AB Canada: ACM; 2020. P. 138–149. https://doi.org/10.1145/3358960.3379124
3. Curino C., Godwal N., Kroth B., Kuryata S., Lapinski G., Liu S., et al. MLOS: An infrastructure for automated software performance engineering. In: *Proceedings of the Fourth International Workshop on Data Management for End-to-End Machine Learning.* 2020:1–5. https://doi.org/10.1145/3399579.3399927
4. Jiang Z.M., Hassan A.E. A survey on load testing of large-scale software systems. *IEEE Transactions on Software Engineering.* 2015;41(11):1091–1118. https://doi.org/10.1109/TSE.2015.2445340
5. Alankar B., Sharma G., Kaur H., Valverde R., Chang V. Experimental setup for investigating the efficient load balancing algorithms on virtual cloud. *Sensors.* 2020;20(24):7342. https://doi.org/10.3390/s20247342
6. Spanaki P., Sklavos N. Cloud Computing: security issues and establishing virtual cloud environment via Vagrant to secure cloud hosts. In: Daimi K. (Ed.). *Computer and Network Security Essentials*. Springer, Cham; 2018. P. 539–553. https://doi.org/10.1007/978-3-319-58424-9_31
7. Saingre D., Ledoux T., Menaud J.-M. BCTMark: a framework for benchmarking blockchain technologies. In: *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications* (*AICCSA*). Antalya, Turkey: IEEE; 2020. P. 1–8. https://doi.org/10.1109/AICCSA50499.2020.9316536
8. Potdar A.M., Narayan D.G., Kengond S., Mulla M.M. Performance evaluation of Docker container and virtual machine. *Procedia Computer Science.* 2020;171:1419–1428. https://doi.org/10.1016/j.procs.2020.04.152

9. Kucek S., Leitner M. An empirical survey of functions and configurations of open-source Capture the Flag (CTF) environments. *J. Network Comput. Appl.* 2020;151:102470. https://doi.org/10.1016/j.jnca.2019.102470
10. Chirigati F., Rampin R., Shasha D., Freire J. ReproZip: Computational reproducibility with ease. In: *Proceedings of the 2016 International Conference on Management of Data*. New York, USA: Association for Computing Machinery; 2016. P. 2085–2088. https://doi.org/10.1145/2882903.2899401
11. Steeves V., Rampin R., Chirigati F. Using ReproZip for reproducibility and library services. *IASSIST Quarterly.* 2018;42(1):14–14. https://doi.org/10.29173/iq18
12. Jimenez I., Sevilla M., Watkins N., Maltzahn C., Lofstead J., Mohror K., et al. The Popper convention: making reproducible systems evaluation practical. In: *2017 IEEE International Parallel and Distributed Processing Symposium Workshops* (*IPDPSW*). 2017. P. 1561–1570. https://doi.org/10.1109/IPDPSW.2017.157
13. Papadopoulos A.V., Versluis L., Bauer A., Herbst N., von Kistowski J., Ali-Eldin A., et al. Methodological principles for reproducible performance evaluation in cloud computing. *IEEE Trans. Software Eng.* 2021;47(8): 1528–1543. https://doi.org/10.1109/TSE.2019.2927908
14. Artač M., Borovssak T., Di Nitto E., Guerriero M., Tamburri D.A. DevOps: introducing infrastructure-as-code. In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion* (*ICSE-C*). 2017. P. 497–498. https://doi.org/10.1109/ICSE-C.2017.162
15. Marquardson J. Infrastructure tools for efficient cybersecurity exercises. *Inform. Systems Education. J.* 2018;16(6):23–30.
16. Šimec A., Držanić B., Lozić D. Isolated environment tools for software development. In: *2018 International Conference on Applied Mathematics Computer Science* (*ICAMCS*). 2018. P. 48–484. https://doi.org/10.1109/ICAMCS46079.2018.00016
17. Stillwell M., Coutinho J.G.F. A DevOps approach to integration of software components in an EU research project. In: *Proceedings of the 1st International Workshop on Quality-Aware DevOps*. New York, USA: Association for Computing Machinery; 2015. P. 1–6. https://doi.org/10.1145/2804371.2804372
18. Magomedov S., Ilin D., Nikulchev E. Resource analysis of the log files storage based on simulation models in a virtual environment. *Appl. Sci.* 2021;11(11):4718. https://doi.org/10.3390/app11114718
19. Staubitz T., Brehm M., Jasper J., Werkmeister T., Teusner R., Willems C., et al. Vagrant virtual machines for hands-on exercises in massive open online courses. In: Uskov V.L., Howlett R.J., Jain L.C. (Eds.). *Smart Education and e-Learning 2016*. Springer, Cham; 2016. P. 363–373. https://doi.org/10.1007/978-3-319-39690-3_32
20. Berger O., Gibson J.P., Lecocq C., Bac C. Designing a virtual laboratory for a relational database MOOC. In: *Proceedings of the 7th International Conference on Computer Supported Education*. Lisbon, Portugal: SCITEPRESS – Science and and Technology Publications; 2015. P. 260–268. https://doi.org/10.5220/0005439702600268
21. Hobeck R., Weber I., Bass L., Yasar H. Teaching DevOps: a tale of two universities. In: *Proceedings of the 2021 ACM SIGPLAN International Symposium on SPLASH-E*, New York, USA: Association for Computing Machinery; 2021. P. 26–31. https://doi.org/10.1145/3484272.3484962
22. Shah J., Dubaria D., Widhalm J. A survey of DevOps tools for networking. In: *2018 9th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference* (*UEMCON*). 2018. P. 185–188. https://doi.org/10.1109/UEMCON.2018.8796814
23. Sandobalín J., Insfran E., Abrahão S. On the effectiveness of tools to support infrastructure as code: Model-driven versus code-centric. *IEEE Access.* 2020;8:17734–17761. https://doi.org/10.1109/ACCESS.2020.2966597

About the author

Dmitry Ilin, Cand. Sci. (Eng.), Associate Professor, Department of Data Processing Digital Technologies, Institute of Cybersecurity and Digital Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: i@dmitryilin.com. ResearcherID J-7668-2017, Scopus Author ID 57203848706, https://orcid.org/0000-0002-0241-2733, https://www.researchgate.net/profile/Dmitry-Ilin-2

Об авторе

Ильин Дмитрий Юрьевич, к.т.н., доцент, кафедра «Цифровые технологии обработки данных» Института кибербезопасности и цифровых технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: i@dmitryilin.com. ResearcherID J-7668-2017, Scopus Author ID 57203848706, https://orcid.org/0000-0002-0241-2733, https://www.researchgate.net/profile/Dmitry-Ilin-2

Translated from Russian into English by Evgenii I. Shklovskii
Edited for English language and spelling by Thomas A. Beavitt