mathematics

MDPI

Article

Technology Stack Selection Model for Software Design of Digital Platforms

Evgeny Nikulchev [1,*], Dmitry Ilin [1] and Alexander Gusev [2,3]

[1] Department of Intelligent Information Security Systems, MIREA—Russian Technological University, Moscow 119454, Russia; ilin_dyu@mirea.ru
[2] Data Center, Russian Academy of Education, Moscow 119121, Russia; alexander.gusev@rusacademedu.ru
[3] Kuban State Technological University, Krasnodar 350072, Russia
* Correspondence: nikulchev@mail.ru

Abstract: The article is dedicated to the development of a mathematical model and methodology for evaluating the effectiveness of integrating information technology solutions into digital platforms using virtual simulation infrastructures. The task of selecting a stack of technologies is formulated as the task of selecting elements from sets of possible solutions. This allows us to develop a mathematically unified approach to evaluating the effectiveness of different solutions, such as choosing programming languages, choosing Database Management System (DBMS), choosing operating systems and data technologies, and choosing the frameworks used. Introduced technology compatibility operation and decomposition of the evaluation of the efficiency of the technology stack at the stages of the life cycle of the digital platform development allowed us to reduce the computational complexity of the formation of the technology stack. A methodology based on performance assessments for experimental research in a virtual software-configurable simulation environment has been proposed. The developed solution allows the evaluation of the performance of the digital platform before its final implementation, while reducing the cost of conducting an experiment to assess the characteristics of the digital platform. It is proposed to compare the characteristics of digital platform efficiency based on the use of fuzzy logic, providing the software developer with an intuitive tool to support decision-making on the inclusion of the solution in the technology stack.

Keywords: mathematical model for evaluating the effectiveness of integrating information technology; digital platforms; virtual simulation infrastructures; experimental virtual environment

Citation: Nikulchev, E.; Ilin, D.; Gusev, A. Technology Stack Selection Model for Software Design of Digital Platforms. *Mathematics* **2021**, *9*, 308. https://doi.org/10.3390/math9040308

Academic Editor: Hsien-Chung Wu
Received: 17 December 2020
Accepted: 30 January 2021
Published: 4 February 2021

1. Introduction

The proliferation of web applications, driven by their platform and hardware independence, ubiquity of use, interfaces, data transfer protocols, and programmable capabilities, has defined the development of the IT sector—the creation of digital platforms. Using a platform allows the collection and sharing of information between a huge number of users, combining results into big data. Information technologies, which are used in the development of digital platforms, are commonly called technology stacks. An important feature of IT solutions integrated into the stack is their replaceability, meaning one of the technologies can be replaced with an alternative, either newly created or a new version of the existing one. There are many techniques for individual software design phases [1,2] for specific technologies and software systems such as digital platforms.

The system performance depends on the efficiency of each of the components of the technology stack [3] and on the effectiveness of their interaction [4]. At the same time, there can be more than one ready-made technology solution for one task, both commercial and free of charge. In practice, the choice is based on load tests or expert assessments. The approaches summarize the experience of using specific components or the technology stack, but are not based on formal assessments and cannot be used to compare efficiency. Formal methods are focused on solving identification and optimization tasks that are of greater

dimensionality. The proposed methods do not consider the specifics of the operation of the digital platform and its infrastructure.

The quality of the technology selection can only be judged after the entire stack of technologies has been formed, a digital platform has been developed, and characteristics are calculated. In practice, there are situations where large digital systems stop working when they start at high load. For example, the logging framework accessed the database where the main data was stored, which resulted in a significant increase in the latency of access to the data. It is typical that when the system is commissioned, it turns out that some of its write/reading functions are slower than expected, the method of storing data was incorrectly selected, and so on. In high load systems with integrated modules used, it is difficult or impossible to assess theoretically their effectiveness. In these conditions, a model and an approach are proposed, which consist of identifying a subset of technologies of the required technology stack and choosing based on an assessment of quality characteristics in conditions that simulate a real environment, e.g., parameters of network loading, parameters of virtual machines, data transfer rate, and so on.

The paper also proposes use of fuzzy logic. For example, when choosing technological solutions based on minimizing the consumption of a resource, the key indicator is not the specific number of bytes or microseconds spent on the execution of an algorithm that changes slightly from experiment to experiment, but a qualitative estimate of whether resource consumption is "high", "medium", or "low" in accordance with the developer's goals and perceptions. The introduction of such quality categories makes it possible to significantly simplify the evaluation of the technological solution, breaking all the many available technological solutions into a small number of classes relative to the consumption of the resource, corresponding to the quality categories.

The article consists of six sections. Section 2 provides an overview of related works, Section 3 proposes a basic model, Section 4 describes the virtual environment used for experimental research, and Section 5 describes the example of decision-making for specific experimental studies. Section 6 provides key results and conclusions.

2. Related Works

The basics of evaluating information technology solutions are considered within the algorithmic efficiency theory. However, in the development of digital platforms, each solution can include a large number of algorithms. Evaluating the effectiveness of each individual algorithm will require laborious research. In addition, the solutions under consideration may contain closed source code.

There are many approaches to the task of selecting effective software components [5–7], methods for solving the problem of optimization, and formal models for the decision-making support [8,9]. However, the tasks under consideration are of great dimensionality and the existing solutions do not take into account the specifics of the operation of the digital platform and its infrastructure. Various methods of Database Management System (DBMS) benchmarking are well known for SQL, NoSQL, and hybrid solutions, but these methods do not address the DBMS in the context of the technology stack.

Development practices (load testing, benchmarking, expert review etc.) generalize experience in specific solutions or technology stacks and are actively applied in digital platform development practices. A significant drawback of these practices is the lack of consideration of the specifics of the operation of the digital platform and its infrastructure.

The need for experimental evaluation of the technology solutions before integration into the digital platform can be due to various reasons. For example, the software developer may need to test the hypothesis about the pros or cons of the solution under consideration [10]. The need for cross-platform functioning [11] also can be the reason for experimental evaluation to ensure the resulting digital platform components can run in various environments (browsers, mobile devices, operating systems etc.).

Distributed software development teams have a practice of using virtual development environments [12]. This technology uses virtualization and virtual machine configura-

tion management [13] to apply the right settings and install the required components. It automates the process of synchronizing, setting up, and starting the developer's work environment. Configuration management systems facilitate simultaneous distributed work on multiple components of the software being developed [14] and automate the process of installing and customizing all the necessary components of the development environment [15]. Upgrading and modifying the virtual environment also becomes simplified [16] because configuration files can be easily distributed among developers.

To test the compatibility of technology solutions, services are used to bootstrap a virtual machine with the chosen operating system, software, and browser of a given version. Microsoft has a number of virtual machines for Internet Explorer and Edge browsers that do not require the purchase of a Windows license. However, since there are many other browsers, there are tools with a large set of options. For example, BrowserStack provides the ability to run virtual machines with a predetermined configuration on a remote server. It also provides means to run automated test scenarios, such as regression testing, during development.

A well-established approach to experimental software evaluation is software testing [17]. In papers [18,19], a number of approaches to testing are considered, which differ in their types: functional, non-functional, compatibility, reliability, recoverability, performance, maintainability, security, and others. As noted in [20], there are noticeable differences in views on the problems of software testing in industry and in science. At the stage of software maintenance, automation tools are actively used [21,22]. Testing the software is commonly included into the overall sequence of operations required to verify that the software meets the requirements. In addition, experiments are being carried out to assess the quality of the system [23,24] for the end users.

3. Technology Stack Selection Model

The concept and corresponding model for choosing a technology stack have been developed and can be described as follows. It is necessary to construct a p-dimensional vector of the technology stack to build the digital platform:

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_p\}, \xi_i \in \hat{S}_i, \ (i = \overline{1, p}),$$

where ξ_i—information technology solution for the technology stack (communication protocols, type of DBMS, frameworks used, Operating system (OS) version, etc.); $\hat{S}_i$—the set of possible alternatives for each information technology solution type. Let each set with specific selected technology options denote Ξ_q.

For the given operating conditions (that is, when the digital platform is used after its complete implementation, during the workload on software and hardware of a computing system), each set of the technology stack can be associated with a vector of efficiency characteristics, such as memory consumption, processing time of a given number of records, processing queue size, failure frequency, maximum number of users, average CPU load, client data transfer time, etc.:

$$\forall \Xi_q \to \vartheta_q = [\varphi_{q,1}, \ldots, \varphi_{q,n}]^{\mathrm{T}} \in \mathbb{R}^n.$$

The stack Ξ_o will be effective if

$$\exists \Xi_o, \ \forall \Xi_l, \ l \neq o : \max_{\Omega}(\vartheta_l, \vartheta_o) = \vartheta_o.$$

Here, Ω is the configuration and operating conditions of the digital platform after its implementation; max—operation of comparison of vectors characterizing qualities. In this paper, the operation max is proposed to be implemented using fuzzy logic.

In this case, the choice is a problem with computational complexity determined by a complete enumeration of all elements of the sets $\hat{S}_i(i = \overline{1,p})$ in p places of the technology stack, i.e., it is necessary to enumerate all the options.

$$O = n_1 \cdot n_2 \cdot \ldots \cdot n_p,$$

where n_i is cardinality of $\hat{S}_i(i = \overline{1,p})$.

The complexity of the problem lies in the necessity of complete enumeration of possible solutions, and in the fact that efficiency can be determined only by forming the entire technology stack and assessing its performance after implementation. It is proposed to solve the problem of evaluating the effectiveness by its approximation on the basis of experimental virtual environments simulating the given operating conditions, decomposing the general problem in accordance with the stages of the life cycle of the development of digital platforms. To achieve this goal, the concept of a configuration is introduced at t stages of the life cycle for a given configuration and operating conditions Ω: $\omega^i(i = \overline{1,t}) \subset \Omega$. At each stage ω^i, each information technology solution of the technology stack is selected so that the values of the efficiency characteristics are greater or equal in a given set of alternatives.

In addition, technologies depend on the selection of previous information technology solutions included into the technology stack. For example, the programming languages chosen at the first stage limit the sets of libraries, the choice of the type of data storage limits the choice of DBMS, and the choice of the OS also introduces restrictions.

Let an operation of compatibility of solutions be introduced such that

$$\xi_k \rhd\lhd \xi_g, \text{ if } \Xi = (\xi_1, \ldots, \xi_k, \ldots, \xi_g, \ldots, \xi_p) \to \vartheta,$$
$$\text{wherein } \vartheta \text{ has no zero elements,}$$

In other words, compatible solutions are those that do not give zero efficiency values; that is, they are able to function when used together.

At each m-th stage of the life cycle, the problem of choosing an information technology solution for the formation of a technology stack is solved, i.e., a subset of the required Ξ_o is formed. The procedure for choosing an information technology solution is as follows. For each valid and compatible set $\Sigma_m = \left\{s_1^m, \ldots, s_\eta^m\right\}, \eta < p$.

$$\begin{array}{l}
\text{Let } s_1^m \in \hat{S}_1, \\
\text{then} \\
s_2^m \in \widetilde{S}_2 \subseteq \hat{S}_2,\ \forall \widetilde{s}_2 \in \widetilde{S}_2 :\ \widetilde{s}_2 \rhd\lhd s_1^m; \\
s_3^m \in \widetilde{S}_3 \subseteq \hat{S}_3,\ \forall \widetilde{s}_3 \in \widetilde{S}_3 :\ \widetilde{s}_3 \rhd\lhd s_2^m; \\
\ldots \\
s_\eta^m \in \widetilde{S}_\eta \subseteq \hat{S}_\eta,\ \forall \widetilde{s}_\eta \in \widetilde{S}_\eta :\ \widetilde{s}_\eta \rhd\lhd s_{\eta-1}^m;
\end{array}$$

For the efficiency vector given at the m-th stage out of M^i characteristics $\forall \Sigma_m \to \Phi_m \in \mathbb{R}^{M^m}$:

$$\exists \Sigma_o,\ \forall \Sigma_l,\ l \neq o :\ \max_{\omega^m}(\Phi_l, \Sigma_o) = \Sigma_o, \text{ for } \omega_i \text{ obtaining a solution set for the technology stack } \Sigma_o \subset \Xi_o.$$

When choosing solutions with the introduced operation of compatibility of information technology solutions, the number of options for enumeration is reduced, so the complexity estimate will be

$$\widetilde{O} = n_1 \cdot n_2(1 - \Delta_1) \cdot n_3(1 - \Delta_2) \cdot \ldots \cdot n_p(1 - \Delta_{p-1}),$$

where n_i is the set cardinality $\hat{S}_i(i = \overline{1,p})$; $\Delta_i(i = \overline{1,\ p-1},\ \Delta_1 \leq \Delta_2 \leq .. \leq \Delta_{p-1})$ is the coefficient characterizing the decrease in cardinality $\hat{S}_i$ down to $\widetilde{S}_i$, considering solution compatibility.

The original problem of evaluating a technology stack is divided into stages. Thus, the technology stack is evaluated at each stage instead of a single evaluation after the digital platform is launched. If the completely assembled digital platform does not meet the requirements (in terms of speed, resources used, the ability to provide desired Quality-of-service (QoS) to users, etc.), it will be necessary to identify which of the technology solutions used affect efficiency (which is a very time-consuming task), and it will be necessary to reimplement or to replace these technology solutions. When using the proposed approach, assessments of the effectiveness of various alternative options are obtained at the stage of selecting technological solutions. These assessments make it possible to select effective and appropriate technological solutions before the time the platform is put into operation.

Thus, the approach allows the reduction of the number of options required to be evaluated for forming a technology stack and makes it possible to evaluate information technology solutions at the stages of the digital platforms' development life cycle. The introduced decomposition of the selection problem allows us to reduce the dimension of the original problem and reduce the number of options under consideration, the effectiveness of which can only be assessed by conducting experiments including each of the information technology solutions into the digital platform.

The approach has limitations that must be considered when using it. The initial selection of the information technology solutions is carried out with the involvement of expert assessments, and therefore the list of options may not be complete. If initial expert assessments have led to an ineffective set of solutions, then the choice of subsequent solutions for implementation in the technology stack will be limited by the need to ensure compatibility with existing ineffective solutions. Thus, a systematic error in expert assessments can hypothetically lead to a decrease in the efficiency of the digital platform.

4. Experimental Virtual Environment

When setting up an experiment, it is important to minimize the influence of the observer on the object. To isolate the evaluated information technology solution from the influence of the observer, it is necessary to form an independent infrastructure [25]. It can be prepared both in hardware, using physical computing devices (computers, servers, routers, etc.), and software, using virtual machines. The second option should be considered preferable, since the implementation of the infrastructure using software means more rational use of resources and portability. In addition, the use of virtual machines provides infrastructure reusability.

It should be noted that infrastructure provisioned with virtual machines has several disadvantages—it requires a large amount of disk space, it is difficult to monitor the current state of virtual machines, and the changes you make need to be documented separately.

To mitigate the shortcomings, one can use the "infrastructure as code" approach. The approach is implemented using systems such as:

- Systems for creating and configuring a virtual development environment (for example, the Vagrant system);
- Systems for automating the deployment and management of applications in environments with containerization support (for example, Docker);
- Configuration management systems (for example, Ansible, Puppet).

Studies show that containerization systems are less suitable for setting up computational experiments. They provide less isolation of computational resources, which can affect results.

To obtain experimental evaluations of the integration of information technology solutions into digital platforms, a virtual simulation bench has been developed, as shown in Figure 1.

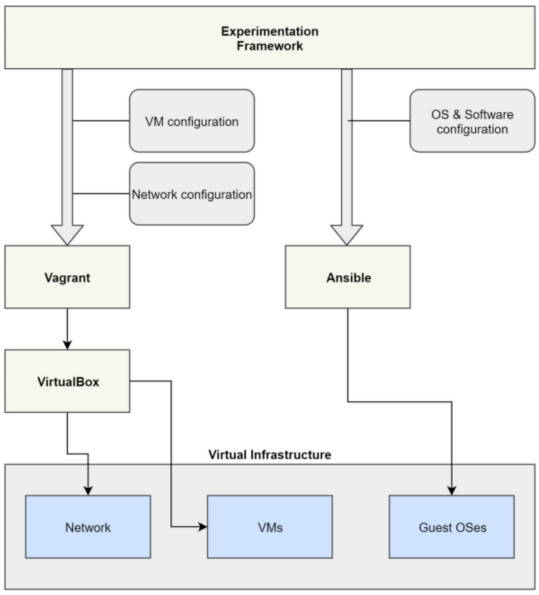

Figure 1. Scheme of the virtual bench for obtaining experimental evaluations of the integration of information technology solutions.

Figure 1 contains a general scheme of the experimental bench. Configuration files in YAML and Ruby languages are used as initial data. Based on the configuration files, virtual machines are launched with the specified parameters and network connection settings. Then the reference image of the operating system is loaded and launched. After launch, the necessary software is installed on the guest operating systems using the configuration management system (Ansible). The detailed description of the experiment, the source code of the virtual environment, the settings of the swarm intelligence algorithm, and other parameters are presented in the paper [25].

The proposed approach based on a virtual environment allows the obtaining of reliable estimates of the effectiveness of information technology solutions. If the requirements and operating conditions are changed (for example, the computing infrastructure is changed, servers were replaced, the amount of data received, and the number of users were changed), then the reliability of the estimates could be arguable. In this case, the estimates of the effectiveness of information technology solutions need to be recalculated. However, the methodology makes it possible to recalculate the value of the estimates using the experimental virtual environment, even if the requirements and operating conditions are subject to change.

Incorrect decomposition of the technology stack into subsets also can significantly affect the reliability of the estimates obtained. That is, the technology stack can be decomposed so that interrelated and interacting information technology solutions are selected at different stages, while the experimental evaluation of the effectiveness of these solutions occurs independently of each other, which excludes the possibility of testing and evaluating their mutual influence. In this case, the reliability of efficiency estimates can be lower than expected; however, this limitation is general for all decomposition problems. Decomposition of a single problem into many elementary problems increases the speed and reduces computational costs of solving them, but it excludes the possibility of assessing their mutual influence. Therefore, the depth of decomposition of the problem into subsets is determined by the researcher depending on the available time and computational resources for solving problems of assessing the effectiveness of information technology solutions.

5. Examples

Let there be given n functional requirements q_i $(i = \overline{1, n})$ for the digital platform, as well as t different configurations ω^k $(k = \overline{1, t})$ of the infrastructure, reflecting the set of conditions for the functioning of the platform. The platform developer defines M

solutions of the technology stack to choose from. Each of the M solutions implement at least one of the functional requirements q_i. The subsets of alternative information technology solutions from M capable of implementing the functional requirement q_i can be denoted as m_i $(i = \overline{1,n})$. The subsets of information technology solutions, where for each functional requirement q_i there is at least one component from M, are defined as technology stacks s^j $\left(j = \overline{1,p}\right)$. S is the set of all possible stacks. To assess the quality of integration of information technology solutions, f quality indicators $r_\xi^{k,j}$,$(\xi = \overline{1,f})$: $R^{k,j} = [r_1^{k,j}, \ldots, r_\xi^{k,j}, \ldots, r_f^{k,j}]^{\mathrm{T}}$, $(k = \overline{1,\ t}, j = \overline{1,p})$ are introduced [26]. These quality indicators belong to $\mathbb{R}^f$ space. Thus,

$$\forall \omega^k \ : s^j \to R^{k,j} \in \mathbb{R}^f \ ,$$

where $R^{k,j}$ is a vector consisting of the values of the experimentally calculated quality indicators for the infrastructure configuration ω^k and the evaluated stack s^j.

The following methodology for the integral quality assessment of the technology stack is proposed:

1. Mathematical formalization of the problem of choosing the appropriate technology stack in accordance with the above definitions.
2. Formation of fuzzy inference rules based on the goals and priorities of the digital platform developer.
3. Study of the fuzzy inference system for the completeness of coverage of the range of input values by the rules, the absence of redundant rules, and the elimination of the ambiguous choice situation by setting the weights of the rules.
4. The choice of the method of normalizing the values of quality indicators $r_\xi^{k,j}$ for transmission to the input of the fuzzy inference system.
5. Organization of experiments in a virtual simulation environment to obtain normalized quality values $r_\xi^{k,j}$ for transferring to the input of the fuzzy inference system and obtaining an integral quality indicator of the evaluated stack s^j for infrastructure configuration ω^k.
6. To organize a directed search of the s^* technology stack for configuration ω^k, it is proposed to use the swarm intelligence algorithm [27].

Let us consider the application of the methodology on the example of choosing Node.js modules for developing a digital platform DigitalPsyTools [28], designed to provide information support for population and longitudinal psychological research in Russia.

The following functional requirements are imposed on the modules used for data transmission and processing in the digital platform:

- q_1—sequentially check all elements of the array for compliance with the condition and return an array consisting of elements for which the check gave the value "True" (given alternatives: Lodash, Underscore);
- q_2—apply the specified function to all elements of the array, returning a new array consisting of the transformed elements (given alternatives: Lodash, Underscore, native JavaScript);
- q_3—return the first element of the array (given alternatives: Lodash, Underscore);
- q_4—build full path to file or directory based on specified array of path elements (given alternatives: native path module);
- q_5—find and replace a substring in the given string (given alternatives: native JavaScript);
- q_6—zip the transferred file array and return the generated Zip archive (given alternatives: Adm-zip, jszip, zipit);
- q_7—calculate the MD5 hash sum for the specified dataset (given alternatives: Hasha, md5, Ts-md5);
- q_8—read data from file (given alternatives: Fs-extra, native fs module);
- q_9—read the contents of a directory by returning an array of filenames and subdirectories inside the directory (given alternatives: Fs-extra);

- q_{10}—recursively read the contents of a directory and return an array of filenames and subdirectories inside the directory (given alternatives: Recursive-readdir).

Thus, $n = 10$, $p = 216$.

The quality of functioning is evaluated using $f = 3$ quality indicators: ${r_1}^{k,j}$—physical time spent on the experiment, ns; ${r_2}^{k,j}$—microprocessor time spent on executing user code during the experiment, μs; ${r_3}^{k,j}$—increase in the size of memory pages allocated to the experiment process (including heap, code segment, and stack), bytes. During the experiment, the quality indicators were normalized relative to their maximum values in the experiment, taking values on the interval [0; 1].

The use of these quality indicators is explained by the need to choose a technology stack for which resource consumption in terms of increasing the size of memory pages, processor, and physical time of program execution are minimal, which will ensure the best user experience on various desktop and mobile devices.

The Fuzzy Logic Toolbox for MATLAB engineering software package is used for fuzzy inference. It allows us to make the process of creating and configuring fuzzy inference systems interactive. The developer visually configures the number of fuzzy sets, the type of the membership function, the method of fuzzification of the initial quantitative data for the transition to a qualitative representation, and the defuzzification method to obtain a quantitative value at the output of the system. In the given example, the following standard fuzzy inference parameters are set: *and method: min*; *or method: max*; *implication: min*; *aggregation: max*; *defuzzification: centroid*.

Two different fuzzy inference systems of the Mamdani type were developed to obtain integral quality indicators of the evaluated technology stack $\Psi\left(\omega^k, s^j\right)$. Both systems are described in Appendix A.

The first fuzzy inference system led to the following technology stack for the implementation of functional requirements:

- q_1 is implemented by the "Underscore" component;
- q_2 and q_5 are implemented by the JavaScript language tools;
- q_3—by the "Lodash" component;
- q_4—by the "Path" component;
- q_6—by the "Adm-zip" component;
- q_7—by the "Hasha" component;
- q_8—by the "Fs" component;
- q_9—by the "Fs-extra" component;
- q_{10}—by the "Recursive-readdir" component.

The integral quality indicator value for the technology stack is 0.8123.

The second fuzzy inference system led to the following technology stack for the implementation of functional requirements of the given digital platform:

- q_1 is implemented by the "Underscore" component;
- q_2 and q_5 are implemented by the JavaScript language tools;
- q_3—by the "Lodash" component;
- q_4—by the "Path" component;
- q_6—by the "Adm-zip" component;
- q_7—by the "Ts-md5" component;
- q_8—by the "Fs-extra" component;
- q_9—by the "Fs-extra" component;
- q_{10}—by the "Recursive-readdir" component.

The integral quality indicator value for the technology stack is 0.8647.

6. Conclusions

A methodology for the selection of information technology solutions for a technology stack of digital platforms based on fuzzy logic has been developed. The methodology was tested on the choice of a technology stack for the component of data processing and trans-

mission in the digital platform of population psychological research. The obtained results were confirmed experimentally, as the implementation of the selected technologies provided the required level of quality and efficiency in the collection and transmission of data in population studies. In this paper, studies of two alternative methods of fuzzy inference are carried out, demonstrating the use of fuzzy logic for the developed methodology.

The contribution of the paper to the developer community lays in demonstration of the importance of conducting experimental research and obtaining numerical estimates of technology solution efficiency during the process of digital platform development. It is shown that the software system will satisfy the specified requirements if the choice of the technology stack is reasonable. The limitation of the approach is the need to allocate additional computing resources and specialists for experimental research and analysis of the results obtained. However, these costs are justified for digital platforms that process big data or work with a large number of users, since the methodology helps to avoid many of the errors that are commonly detected at the launch stage.

The proposed methodology can be applied in various models of the digital platforms' life cycle. Since correct experiments are time consuming, it is quite possible that the approach is difficult to apply in agile development methodologies with short sprints. The methodology is suited better to the incremental and agile methodologies with a longer sprint or iteration duration as it gives the advantage during the search of the effective information technology solutions based on the previously selected technology stack.

The proposed methodology can be used when choosing technological solutions for the technology stack of modern digital platforms and similar software systems with integrated architecture.

Author Contributions: Conceptualization, E.N.; methodology, E.N., A.G., and D.I.; software, D.I.; validation, A.G.; formal analysis, E.N. and A.G.; investigation, D.I.; resources, D.I.; writing—original draft preparation, E.N., D.I.; writing—review and editing, A.G.; visualization, A.G.; supervision, E.N.; project administration, E.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Higher Education of the Russian Federation.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Two different fuzzy inference systems of the Mamdani type (hereinafter FIS) were developed, as shown in Figure A1, to obtain integral quality indicators $\Psi\left(\omega^k, s^j\right)$ of the evaluated technology stack. These systems take as input the three quality indicators described above, with ${r_1}^{k,j}$ being denoted as **t**, ${r_2}^{k,j}$ being denoted as **cpu**, and ${r_3}^{k,j}$ being denoted as **ram**. The integral quality indicator $\Psi(\omega^k, s^j)$ is denoted as **quality**.

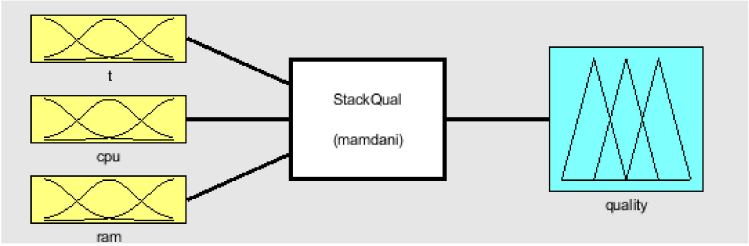

Figure A1. Structure of fuzzy inference system.

In both fuzzy inference systems, the numerical values of the indicators **t**, **cpu**, **ram** are associated with fuzzy sets "low"—low resource consumption, "med"—average resource consumption, "high"—high resource consumption, for which triangular membership functions are defined with coordinates vertices [−0.4 0 0.4], [0.1 0.5 0.9], [0.6 1 1.4] for "low", "med", "high", respectively.

The integral quality indicator named **quality** is associated with fuzzy sets "low"—low quality, "med"—medium quality, "high"—high quality in accordance with defined membership functions, as shown in Figure A2.

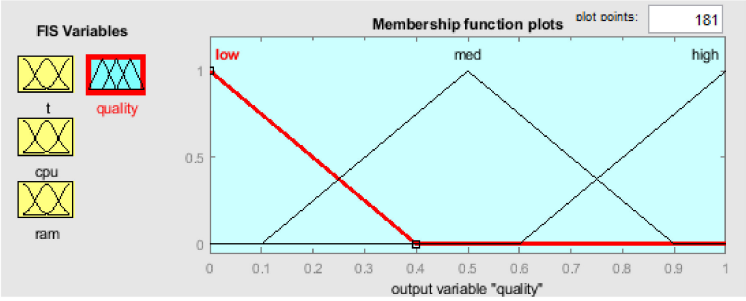

Figure A2. Membership functions of fuzzy sets for the integral quality indicator.

When determining the value of the integral quality indicator in FIS, the following rules are used (the weight of the rule is indicated in brackets):

1. If (**t** is low) and (**cpu** is low) then (**quality** is high) (0.5);
2. If (**t** is low) and (**cpu** is med) then (**quality** is high) (0.5);
3. If (**t** is low) and (**cpu** is high) then (**quality** is med) (0.5);
4. If (**t** is med) and (**cpu** is low) then (**quality** is high) (0.5);
5. If (**t** is med) and (**cpu** is med) then (**quality** is med) (0.5);
6. If (**t** is med) and (**cpu** is high) then (**quality** is med) (0.5);
7. If (**t** is high) and (**cpu** is low) then (**quality** is med) (0.5);
8. If (**t** is high) and (**cpu** is med) then (**quality** is med) (0.5);
9. If (**t** is high) and (**cpu** is high) then (**quality** is med) (0.5);
10. If (**ram** is med) then (**quality** is med) (1);
11. If (**ram** is high) then (**quality** is low) (1).

For verification, an alternative output is considered, using more rules. When determining the value of the integral quality indicator in FIS_A, the following rules are used (the weight of the rule is indicated in brackets):

1. If (**t** is low) and (**cpu** is low) and (**ram** is low) then (**quality** is high) (1);
2. If (**t** is low) and (**cpu** is low) and (**ram** is med) then (**quality** is high) (1);
3. If (**t** is low) and (**cpu** is low) and (**ram** is high) then (**quality** is med) (1);
4. If (**t** is low) and (**cpu** is med) and (**ram** is low) then (**quality** is high) (1);
5. If (**t** is low) and (**cpu** is med) and (**ram** is med) then (**quality** is med) (1);
6. If (**t** is low) and (**cpu** is med) and (**ram** is high) then (**quality** is low) (1);
7. If (**t** is low) and (**cpu** is high) and (**ram** is low) then (**quality** is med) (1);
8. If (**t** is low) and (**cpu** is high) and (**ram** is med) then (**quality** is low) (1);
9. If (**t** is low) and (**cpu** is high) and (**ram** is high) then (**quality** is low) (1);
10. If (**t** is med) and (**cpu** is low) and (**ram** is low) then (**quality** is high) (1);
11. If (**t** is med) and (**cpu** is low) and (**ram** is med) then (**quality** is med) (1);
12. If (**t** is med) and (**cpu** is low) and (**ram** is high) then (**quality** is low) (1);
13. If (**t** is med) and (**cpu** is med) and (**ram** is low) then (**quality** is med) (1);
14. If (**t** is med) and (**cpu** is med) and (**ram** is med) then (**quality** is med) (1);
15. If (**t** is med) and (**cpu** is med) and (**ram** is high) then (**quality** is low) (1);
16. If (**t** is med) and (**cpu** is high) and (**ram** is low) then (**quality** is med) (1);
17. If (**t** is med) and (**cpu** is high) and (**ram** is med) then (**quality** is med) (1);
18. If (**t** is med) and (**cpu** is high) and (**ram** is high) then (**quality** is low) (1);

19. If (**t** is high) and (**cpu** is low) and (**ram** is low) then (**quality** is med) (1);
20. If (**t** is high) and (**cpu** is low) and (**ram** is med) then (**quality** is low) (1);
21. If (**t** is high) and (**cpu** is low) and (**ram** is high) then (**quality** is low) (1);
22. If (**t** is high) and (**cpu** is med) and (**ram** is low) then (**quality** is med) (1);
23. If (**t** is high) and (**cpu** is med) and (**ram** is med) then (**quality** is med) (1);
24. If (**t** is high) and (**cpu** is med) and (**ram** is high) then (**quality** is low) (1);
25. If (**t** is high) and (**cpu** is high) and (**ram** is low) then (**quality** is med) (1);
26. If (**t** is high) and (**cpu** is high) and (**ram** is med) then (**quality** is low) (1);
27. If (**t** is high) and (**cpu** is high) and (**ram** is high) then (**quality** is low) (1).

FIS and FIS_A decision surfaces are shown in Figure A3.

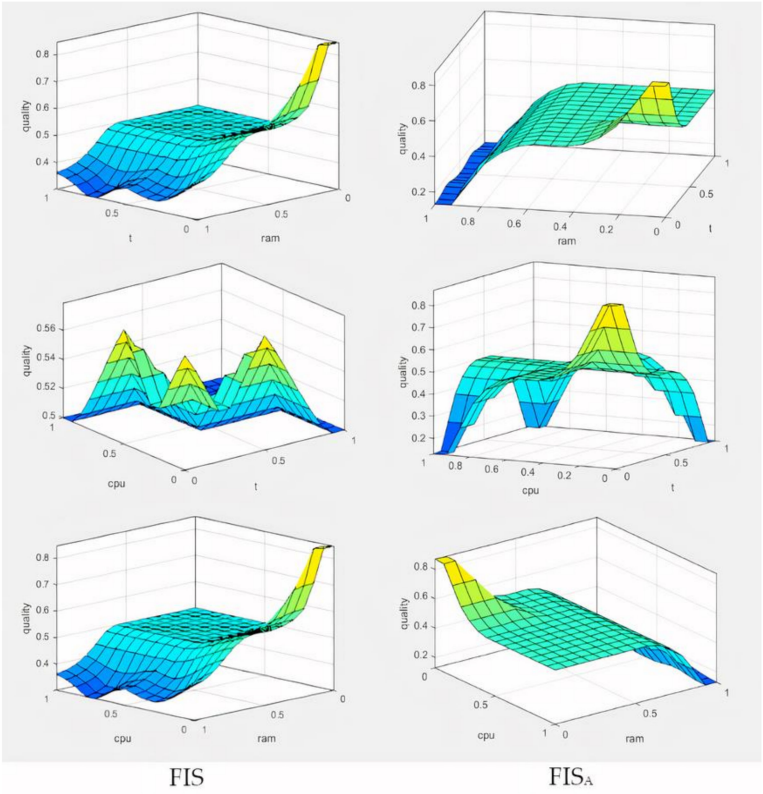

Figure A3. Decision surfaces for fuzzy inference systems, FIS and FIS_A.

Consideration of decision surfaces for pairs of indicators for FIS and FIS_A indicates the applicability of both fuzzy inference systems for the choice of information technology solutions. However, due to a more compact rule base and the use of weight priorities, FIS is distinguished by a higher steepness of the surface in terms of **t**, **cpu** and by the presence of local maximum, which is compensated by the superior weight of the selection rules in terms of the **ram** indicator (Rules 10 and 11) to eliminate the ambiguity of the choice in terms of **t**, **cpu**.

References

1. Ramirez, A.; Romero, J.R.; Ventura, S. Interactive multi-objective evolutionary optimization of software architectures. *Inf. Sci.* **2018**, *463*, 92–109. [CrossRef]
2. Yang, Y.; Yang, B.; Wang, S.; Jin, T.; Li, S. An enhanced multi-objective grey wolf optimizer for service composition in cloud manufacturing. *Appl. Soft Comput.* **2020**, *87*, 106003. [CrossRef]
3. Gholamshahi, S.; Hasheminejad, S.M.H. Software component identification and selection: A research review. *Softw. Pract. Exp.* **2019**, *49*, 40–69. [CrossRef]
4. Beran, P.P.; Vinek, E.; Schikuta, E. A cloud-based framework for QoS-aware service selection optimization. In Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services, Ho Chi Minh City, Vietnam, 5–7 December 2011; pp. 284–287.

5. Ramírez, A.; Parejo, J.A.; Romero, J.R.; Segura, S.; Ruiz-Cortés, A. Evolutionary composition of QoS-aware web services: A many-objective perspective. *Expert Syst. Appl.* **2017**, *72*, 357–370. [CrossRef]
6. Vinek, E.; Beran, P.P.; Schikuta, E. A dynamic multi-objective optimization framework for selecting distributed deployments in a heterogeneous environment. *Procedia Comput. Sci.* **2011**, *4*, 166–175. [CrossRef]
7. Kudzh, S.A.; Tsvetkov, V.Y.; Rogov, I.E. Life cycle support software components. *Russ. Technol. J.* **2020**, *8*, 19–33. [CrossRef]
8. Ezenwoke, A.; Daramola, O.; Adigun, M. QoS-based ranking and selection of SaaS applications using heterogeneous similarity metrics. *J. Cloud Comput.* **2018**, *7*, 15. [CrossRef]
9. Belov, V.; Tatarintsev, A.; Nikulchev, E. Choosing a Data Storage Format in the Apache Hadoop System Based on Experimental Evaluation Using Apache Spark. *Symmetry* **2021**, *13*, 195. [CrossRef]
10. Beyer, D.; Lemberger, T. Software verification: Testing vs. model checking. In *Haifa Verification Conference*; Springer: Cham, Switzerland, 2017; pp. 99–114.
11. Yigitbas, E.; Anjorin, A.; Jovanovikj, I.; Kern, T.; Sauer, S.; Engels, G. Usability evaluation of model-driven cross-device web user interfaces. In *International Conference on Human-Centred Software Engineering*; Springer: Cham, Switzerland, 2018; pp. 231–247.
12. Caballer, M.; Blanquer, I.; Moltó, G.; De Alfonso, C. Dynamic management of virtual infrastructures. *J. Grid Comput.* **2015**, *13*, 53–70. [CrossRef]
13. Giannakopoulos, I.; Konstantinou, I.; Tsoumakos, D.; Koziris, N. Cloud application deployment with transient failure recovery. *J. Cloud Comput.* **2018**, *7*, 1–20. [CrossRef]
14. Xuan, N.P.N.; Lim, S.; Jung, S. Centralized management solution for vagrant in development environment. In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, Beppu, Japan, 5–7 January 2017. [CrossRef]
15. Peacock, M. *Creating Development Environments with Vagrant*; Packt Publishing Ltd.: Birmingham, UK, 2015.
16. Iuhasz, G.; Pop, D.; Dragan, I. Architecture of a scalable platform for monitoring multiple big data frameworks. *Scalable Comput.* **2016**, *17*, 313–321. [CrossRef]
17. Garousi, V.; Giray, G.; Tüzün, E.; Catal, C.; Felderer, M. Aligning software engineering education with industrial needs: A meta-analysis. *J. Syst. Softw.* **2019**, *156*, 65–83. [CrossRef]
18. Lemos, O.A.L.; Silveira, F.F.; Ferrari, F.C.; Garcia, A. The impact of Software Testing education on code reliability: An empirical assessment. *J. Syst. Softw.* **2018**, *137*, 497–511. [CrossRef]
19. Nachiyappan, S.; Justus, S. Cloud testing tools and its challenges: A comparative study. *Procedia Comput. Sci.* **2015**, *50*, 482–489. [CrossRef]
20. Garousi, V.; Felderer, M. Worlds apart: Industrial and academic focus areas in software testing. *IEEE Softw.* **2017**, *34*, 38–45.
21. Couto, L.D.; Tran-Jørgensen, P.W.V.; Nilsson, R.S.; Larsen, P.G. Enabling continuous integration in a formal methods setting. *Int. J. Softw. Tools Technol. Transf.* **2020**, *2*, 667–683. [CrossRef]
22. Mäntylä, M.V.; Adams, B.; Khomh, F.; Engström, E.; Petersen, K. On rapid releases and software testing: A case study and a semi-systematic literature review. *Empir. Softw. Eng.* **2015**, *20*, 1384–1425. [CrossRef]
23. Lindgren, E.; Münch, J. Raising the odds of success: The current state of experimentation in product development. *Inf. Softw. Technol.* **2016**, *77*, 80–91. [CrossRef]
24. Dingsøyr, T.; Lassenius, C. Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Inf. Softw. Technol.* **2016**, *77*, 56–60. [CrossRef]
25. Gusev, A.; Nikulchev, E.; Ilin, D. The Dataset of the Experimental Evaluation of Software Components for Application Design Selection Directed by the Artificial Bee Colony Algorithm. *Data* **2020**, *5*, 59. [CrossRef]
26. Brondolin, R.; Ferroni, M.; Santambrogio, M. Performance-aware load shedding for monitoring events in container based environments. *ACM Sigbed Rev.* **2019**, *16*, 27–32. [CrossRef]
27. Gusev, A.; Ilin, D.; Kolyasnikov, P.; Nikulchev, E. Effective selection of software components based on experimental evaluations of quality of operation. *Eng. Lett.* **2020**, *28*, 420–427.
28. Nikulchev, E.; Ilin, D.; Silaeva, A.; Kolyasnikov, P.; Belov, V.; Runtov, A.; Pushkin, P.; Laptev, N.; Alexeenko, A.; Magomedov, S.; et al. Digital Psychological Platform for Mass Web-Surveys. *Data* **2020**, *5*, 95. [CrossRef]