Russian Technological Journal.
2022;10(5):16–27

ISSN 2782-3210 (Print)
ISSN 2500-316X (Online)

Информационные системы. Информатика. Проблемы информационной безопасности

Information systems. Computer sciences. Issues of information security

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

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.

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.

ВВЕДЕНИЕ

Разработка программного обеспечения (ПО) в значительной степени основывается на использовании готовых информационно-технологических решений. При этом проводится интеграция программных библиотек, фреймворков, систем управления базами данных или целых программных продуктов.

Интегрируемые решения могут обладать различными характеристиками качества, в т.ч. в разных условиях функционирования [1–3]. Оценка характеристик качества, например, производительности [2–4], особенно важна при проектировании систем, обслуживающих большое количество пользователей. Для этого рассматриваемые информационно-технологические решения или результат их интеграции проверяют экспериментально [3–5]. Как правило, такие эксперименты проводят на облачной инфраструктуре, хотя это не всегда целесообразно. Иногда необходимая инфраструктура может быть организована на рабочих станциях [6, 7] исследователей. В обоих случаях организация инфраструктуры требует подготовки виртуальных машин или контейнеров [8, 9], позволяющих оценивать информационно-технологические решения в изолированной среде.

Исследователи сходятся во мнении, что результаты экспериментальных оценок должны быть воспроизводимы [7, 10–13]. Однако существуют различные подходы к обеспечению воспроизводимости. В одних работах отмечается важность подробного документирования инфраструктуры, протокола проведения эксперимента [12], в других – предлагаются инструменты, позволяющие фиксировать действия ПО на уровне системных вызовов [10, 11]. Также предлагается осуществлять подготовку инфраструктуры не вручную, а с помощью технологий, широко применяемых в сфере DevOps[1] [14–17]. Эти технологии зарекомендовали себя, в т.ч. для исследовательских целей [5, 18], т.к. обеспечивают воспроизводимость экспериментов. Кроме того, они применяются и в педагогической деятельности [19–21] для обеспечения участников образовательного процесса идентичным рабочим окружением.

Тем не менее, в исследованиях отмечается, что для реализации воспроизводимого эксперимента необходимо применение множества программных инструментов [10, 11], что требует от исследователя дополнительных знаний и навыков. Также отмечается, что их использование может увеличивать трудозатраты при постановке эксперимента.

Настоящая работа посвящена анализу инструментов, применяемых для реализации инфраструктуры на рабочих станциях исследователей, и разработке фреймворка для проведения экспериментальных исследований по оценке характеристик информационно-технологических решений в виртуальной среде.

Для достижения поставленной цели необходимо:

- провести анализ задач исследователя при постановке эксперимента, а также определить инструментальные средства, применяемые для их решения;
- на основе проведенного анализа подготовить архитектурные решения, определяющие ключевые характеристики программного фреймворка;
- разработать фреймворк, соответствующий предметной области и архитектурным решениям, а также представить методику его применения.

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОВЕДЕНИЯ ЭКСПЕРИМЕНТОВ ПО ОЦЕНКЕ ИНФОРМАЦИОННО-ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ

Перед тем как переходить к разработке программного фреймворка, следует провести анализ задач исследователя при постановке воспроизводимого эксперимента, а также инструментальных средств, применяемых для их решения.

Проведение экспериментальных исследований по оценке характеристик информационно-технологических решений возможно без использования перечисленных далее программных инструментов. Однако целесообразность применения таких средств обоснована.

Использование виртуальных машин (например, с помощью *VirtualBox*[2] [9, 22]) позволяет проводить эксперименты с использованием меньшего числа устройств, вплоть до проведения эксперимента на единственном физическом устройстве.

Сама виртуальная среда может занимать значительный объем дискового пространства, что усложняет передачу виртуальных машин между исследователями. Также сама по себе виртуальная среда не предоставляет инструментов для формального описания ее конфигурации и конфигурации виртуальных машин. Таким образом, при описании экспериментального стенда постфактум появляется вероятность ошибок при документировании деталей эксперимента.

Это может быть скомпенсировано применением инструментов создания и конфигурирования

[1] DevOps – акроним от development & operations (разработка и эксплуатация) – совокупность практик автоматизации технологических процессов сборки, настройки, развертывания программного обеспечения. [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/, дата обращения: 18.02.2022. [https://www.virtualbox.org/. Accessed February 18, 2022.]

виртуальной среды (например, *Vagrant*[3] [9, 22]). Если виртуальные машины заданы с помощью инструмента конфигурации, то получается, что формальное описание характеристик виртуального стенда может быть определено еще до того, как стенд фактически начнет функционировать. В дополнение, средства облачной конфигурации (например, *Ansible*[4] [22, 23]) позволяют задать настройки ПО виртуальных машин аналогичным образом. Такой подход, именуемый «инфраструктура как код» [14, 23], решает две обозначенные проблемы – уменьшает объем экспериментального стенда при передаче между исследователями и обеспечивает надежность документирования деталей эксперимента.

Несмотря на решение описанных проблем, этого недостаточно для проведения экспериментов. Конфигурация виртуального экспериментального стенда не всегда остается неизменной на протяжении всего эксперимента. Например, ограничения на утилизацию вычислительных ресурсов могут требоваться только на определенных стадиях эксперимента. Вышеперечисленные инструменты не позволяют изменять характеристики стенда, т.к. описывают статическую конфигурацию.

Для проведения программно-управляемого эксперимента также необходим менеджер задач (например, *Gulp*[5]), который бы выполнял команды в заданном порядке. Также важными являются инструменты сбора данных мониторинга утилизации вычислительных ресурсов (например, *Atop*[6]) и средства для подготовки отчетов.

Суммируя вышеизложенное, можно прийти к выводу, что для проведения надежно задокументированного программно-управляемого эксперимента в виртуальной среде необходима подготовка виртуального стенда и решение следующих задач:

1) дизайн экспериментального исследования;
2) подготовка формального описания виртуальных машин;
3) подготовка формального описания конфигурации ПО виртуальных машин;
4) установка средств мониторинга утилизации ресурсов на каждую виртуальную машину (в дополнение к п. 3);
5) подготовка материалов исследования (ИТ-решение, экспериментальные данные и т.д.);
6) разработка и отладка программного кода управления экспериментом;
7) обеспечение корректных ограничений по утилизации ресурсов на различных стадиях эксперимента;
8) экспорт данных мониторинга и построение отчета об утилизации ресурсов.

Перечисленное требует от исследователя знания большого перечня технологий и способов их интеграции. Ситуацию также усложняет то, что разные технологии требуют знания разных языков программирования и разметки данных. Примером может служить перечень, показанный в табл. 1. Важно отметить, что знание перечисленных технологий становится обязательным.

Таблица 1. Пример перечня технологий, необходимых для проведения экспериментов в виртуальной среде

Категория	Инструмент	Язык (программирования, разметки)
Конфигурирование виртуальных машин	*Vagrant*	Ruby[7]
Конфигурирование ПО виртуальных машин	*Ansible*	YAML[8]
Управление экспериментом	*Gulp*, Shell script[9]	JavaScript[10], Bash[11]
Генерация отчетов	*Gulp* + *EJS*[12] + *Plotly.js*[13]	JavaScript, HTML[14], JSON[15]
Экспорт данных мониторинга	*Atop*, Shell script	Bash

Поскольку перед исследователем стоит объемный перечень задач, часть из которых не решается существующими инструментами, целесообразно спроектировать и разработать инструмент, который бы обладал следующими характеристиками:

[3] https://www.vagrantup.com/, дата обращения: 18.02.2022. [https://www.vagrantup.com/. Accessed February 18, 2022.]

[4] https://www.ansible.com/, дата обращения: 18.02.2022. [https://www.ansible.com/. Accessed February 18, 2022.]

[5] https://gulpjs.com/, дата обращения: 18.02.2022. [https://gulpjs.com/. Accessed February 18, 2022.]

[6] https://www.atoptool.nl/, дата обращения: 18.02.2022. [https://www.atoptool.nl/. Accessed February 18, 2022.]

[7] https://www.ruby-lang.org/, дата обращения: 18.02.2022. [https://www.ruby-lang.org/. Accessed February 18, 2022 (in Russ.).]

[8] https://yaml.org/, дата обращения: 18.02.2022. [https://yaml.org/. Accessed February 18, 2022.]

[9] Shell script – скрипт, написанный на языке Bash или аналогичном ему языке. [Shell script is a script written in Bash or a similar language.]

[10] https://262.ecma-international.org/6.0/, дата обращения: 18.02.2022. [https://262.ecma-international.org/6.0/. Accessed February 18, 2022.]

[11] http://www.gnu.org/software/bash/, дата обращения: 18.02.2022. [http://www.gnu.org/software/bash/. Accessed February 18, 2022.]

[12] https://ejs.co/, дата обращения: 18.02.2022. [https://ejs.co/. Accessed February 18, 2022.]

[13] https://plotly.com/javascript/, дата обращения: 18.02.2022. [https://plotly.com/javascript/. Accessed February 18, 2022.]

[14] https://html.spec.whatwg.org/multipage/, дата обращения: 18.02.2022. [https://html.spec.whatwg.org/multipage/. Accessed February 18, 2022.]

[15] https://www.json.org/, дата обращения: 18.02.2022. [https://www.json.org/. Accessed February 18, 2022.]

- возможностью формального описания эксперимента в виде программных конфигураций;
- возможностью применения специфичных программных конфигураций на конкретных стадиях эксперимента;
- способностью сократить число языков программирования, разметки данных и информационных технологий, необходимых для проведения эксперимента.

ПРОЕКТИРОВАНИЕ ФРЕЙМВОРКА

В процессе проектирования фреймворка был проведен анализ требований к программному обеспечению и сформирована отвечающая заданным требованиям программная архитектура.

Так как конфигурация виртуальной машины и параметры ее ПО описываются в статике, то их можно задавать с помощью слабоструктурированных форматов. Такой формат должен быть удобен для чтения и редактирования, поэтому предлагается использовать язык разметки YAML. Этот язык применяется в системе *Ansible* для конфигурирования ПО виртуальных машин и знаком исследователям. Вместо использования языка программирования общего назначения алгоритм выполнения автоматизированных задач также можно задавать в декларативном виде на языке YAML.

Подразумевается, что фреймворк для проведения экспериментальных исследований будет компилировать исходные файлы на языке YAML в целевые конфигурации (например, конфигурацию для инструмента *Vagrant* на языке Ruby), и будет содержать встроенное средство автоматизации выполнения задач. Таким образом, исследователю вместо использования трех различных языков для конфигурирования виртуальных машин, их ПО и выполнения автоматизированных задач достаточно только языка разметки YAML, что показано на рис. 1.

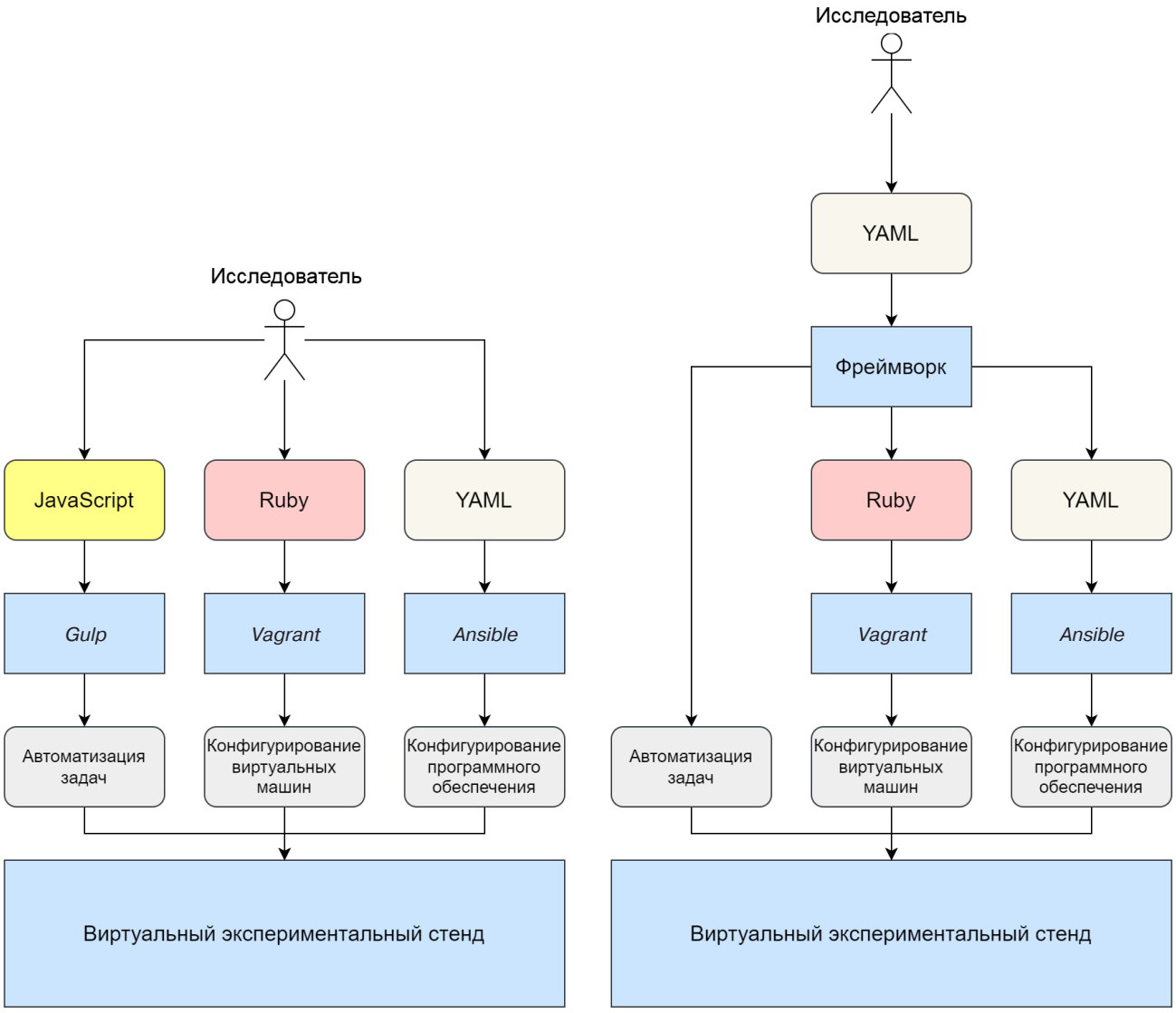

Рис. 1. Сокращение числа применяемых исследователем языков программирования и разметки данных

Для того чтобы обеспечить проведение эксперимента с учетом специфических для каждой стадии конфигураций, предлагается следующее. Эксперимент разделяется на стадии, на каждой из которых к каждой виртуальной машине применяются скомпилированные конфигурации. Они состоят из трех составляющих: (1) общей конфигурации для всего проекта, (2) конфигурации конкретной виртуальной машины и (3) конфигурации, адресованной к определенной стадии эксперимента. После компиляции выбранных конфигураций выполняется ряд действий, задающих алгоритм эксперимента на конкретной стадии. Эти действия могут включать любые команды в отношении виртуальных машин. Диаграмма, иллюстрирующая процесс эксперимента, показана на рис. 2.

Файловая структура проекта для проведения экспериментального исследования показана на рис. 3. Типовой проект состоит из:

- файла конфигурации;
- одной или нескольких виртуальных машин, для каждой из которых также существует отдельный файл конфигурации и набор специфичных для виртуальной машины файлов провизии;
- общих файлов провизии для всех виртуальных машин;
- файлов с исходными данными (включая исследуемое информационно-технологическое решение);
- директорий с отчетами об утилизации ресурсов виртуальных машин в ходе проведенных экспериментов.

Файл конфигурации эксперимента и файлы конфигурации виртуальных машин реализуются в слабоструктурированном формате YAML. Схематично структура этих файлов представлена на рис. 4, где сверху (файл repex.yml) показана конфигурация эксперимента, а снизу (файл vm.yml) – конфигурация одной из виртуальных машин.

Содержимое файлов взаимосвязано. Свойство «path» в конфигурации эксперимента указывает путь до файла vm.yml. Название стадии эксперимента (stage), используемое в файле конфигурации эксперимента, может быть использовано в конфигурации

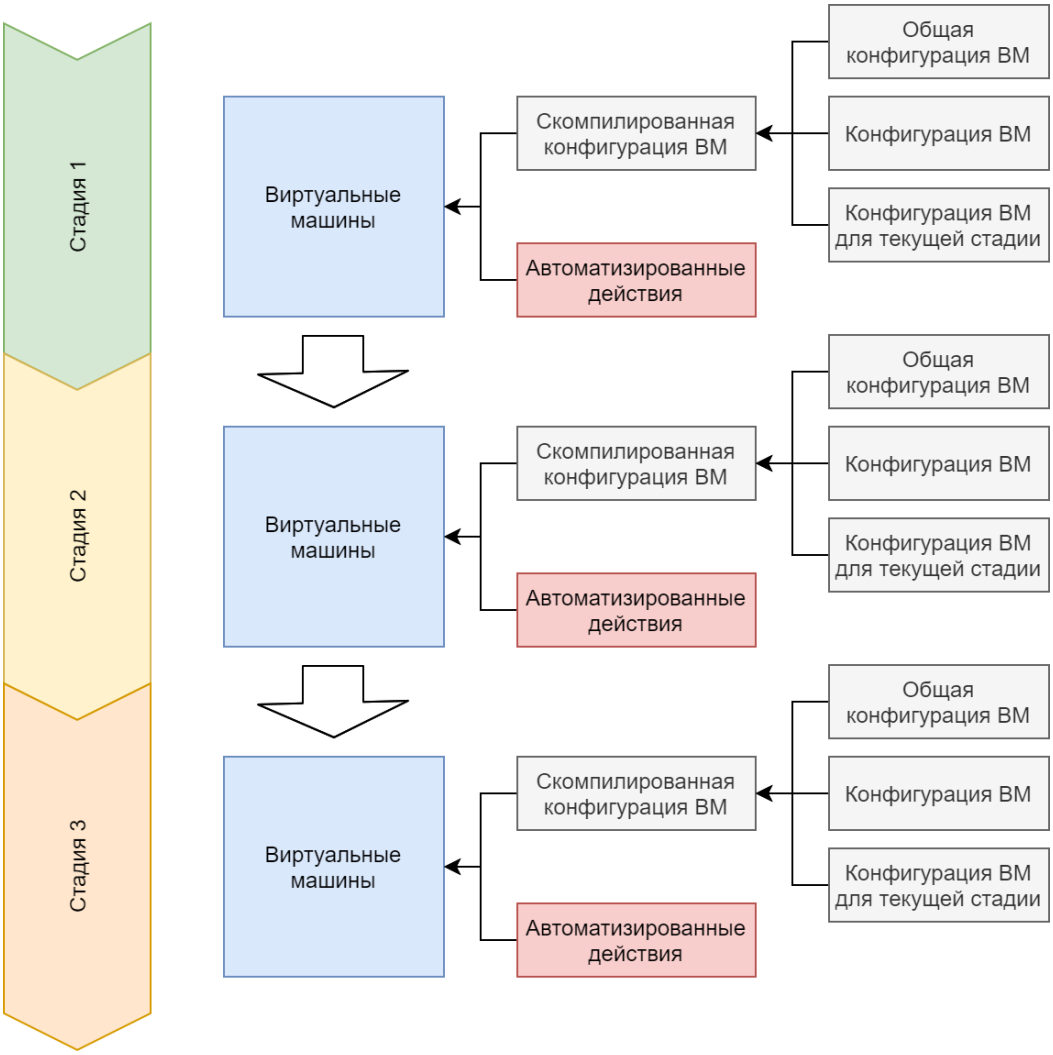

Рис. 2. Диаграмма проведения эксперимента

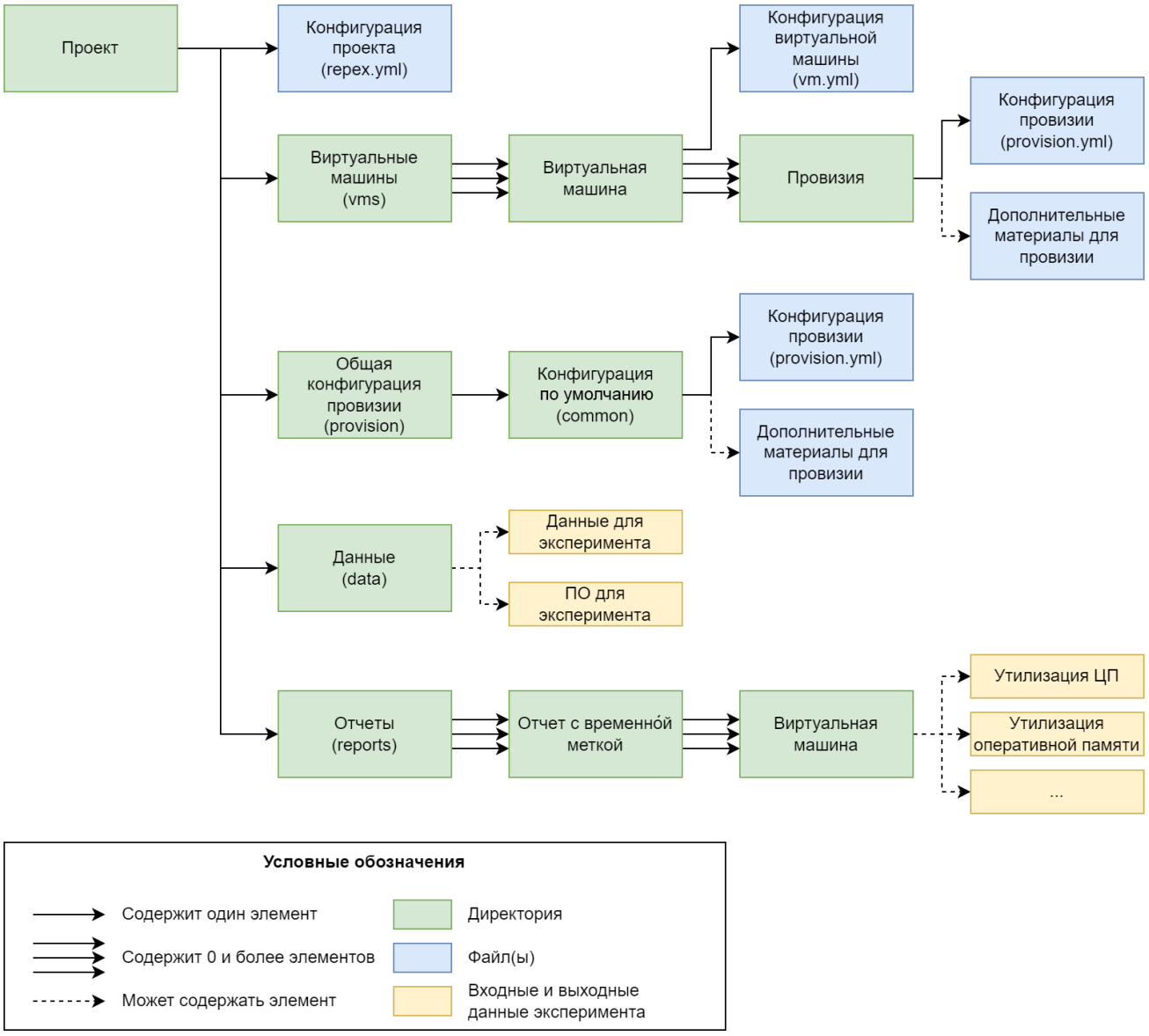

Рис. 3. Файловая структура проекта для проведения экспериментального исследования

виртуальной машины с целью применения ее настроек, отличных от настроек по умолчанию. Эти настройки будут применяться только на указанной стадии эксперимента, однако следует учитывать особенности работы гипервизора. Так, большинство настроек будут применены только после перезагрузки виртуальной машины. Некоторые настройки, такие как ограничения пропускной способности дисковой подсистемы или сетевого канала, применяются на перманентной основе, и потребуется создание еще одной стадии эксперимента для снятия примененных ограничений.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ФРЕЙМВОРКА

На основе полученных результатов проектирования был разработан и опубликован в качестве ПО с открытым исходным кодом программный фреймворк *Repexlab* (reproducible experiment laboratory)[16]. Он является инструментом с интерфейсом командной строки (command line interface, далее – CLI), основная задача которого – поддержка проведения экспериментальных исследований по оценке характеристик ИТ-решений. Фреймворк задает базовую конфигурацию проекта, обеспечивает применение настроек виртуального окружения на различных стадиях эксперимента.

Программный фреймворк базируется на ряде существующих инструментов:

VirtualBox – система виртуализации, которая позволяет организовать работу одной или нескольких виртуальных машин, в том числе на ПК;

[16] https://github.com/rnd-student-lab/repexlab, дата обращения: 18.02.2022. [https://github.com/rnd-student-lab/repexlab. Accessed February 18, 2022.]

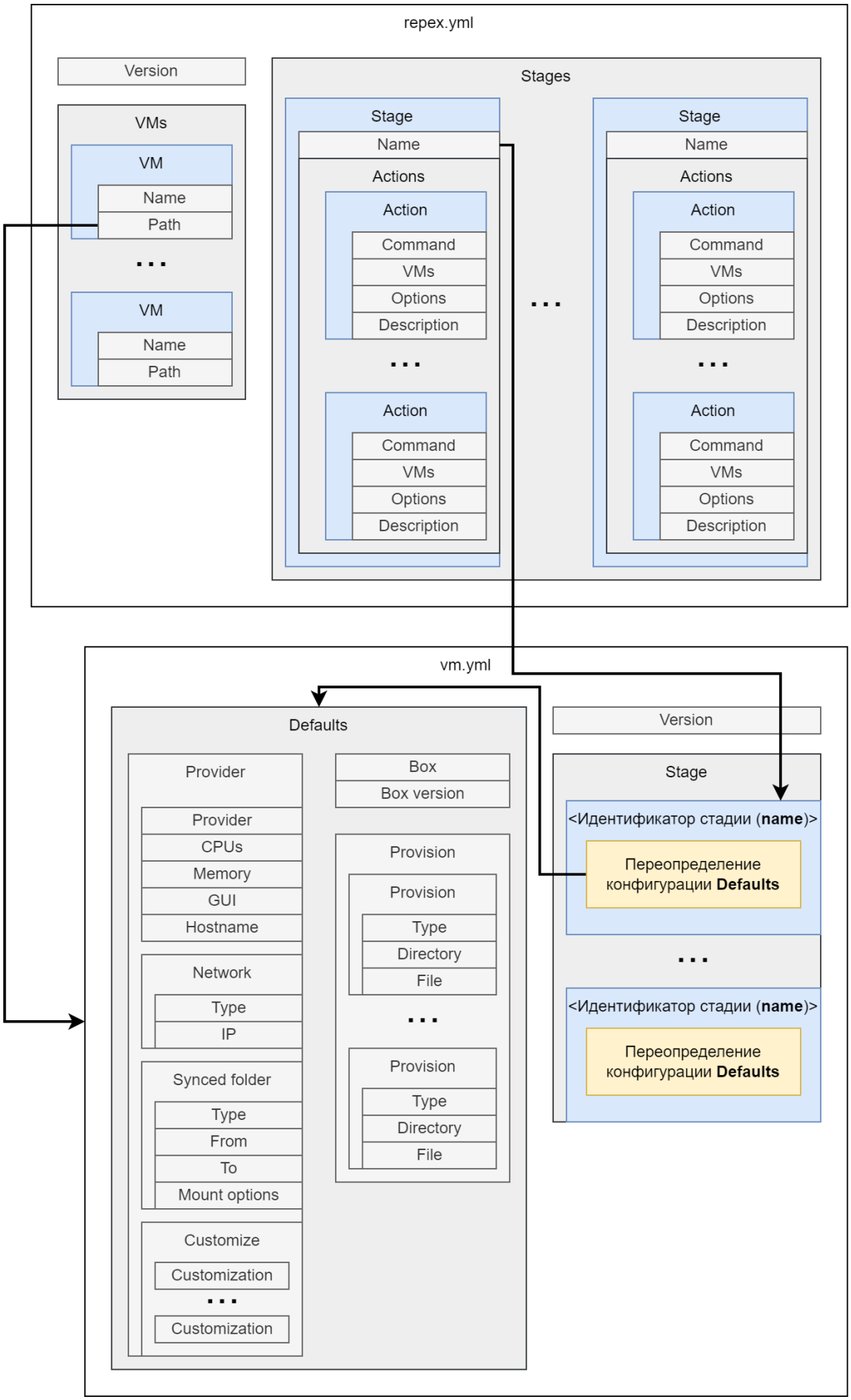

Рис. 4. Связь конфигурационных файлов проекта для проведения экспериментального исследования

Vagrant – инструмент создания и конфигурирования виртуальной среды. Как правило, применяется для подготовки виртуального рабочего окружения разработчиков. В базовом варианте использует *VirtualBox* в качестве гипервизора;

Ansible – инструмент управления облачными конфигурациями. Используется фреймворком как основное средство провизии виртуальных машин;

Atop – инструмент мониторинга процессов в операционной системе. Используется фреймворком для сбора данных о задействовании вычислительных ресурсов виртуальных машин;

Node.js[17] и *NPM*[18] – платформа для исполнения JavaScript-кода и сопутствующий пакетный менеджер. Так как *Repexlab* разработан на JavaScript, они необходимы для установки и исполнения кода фреймворка.

К основным функциям, отличающим фреймворк от совокупности применяемых в нем технологий, относятся:

- создание базового проекта для проведения экспериментального исследования;
- скаффолдинг[19] для управления виртуальными машинами в рамках экспериментального стенда;
- управление виртуальным стендом из командной строки;
- встроенный инструментарий автоматизации выполнения задач взаимодействия с виртуальным стендом.

Разработанный инструмент имеет команды, которые можно использовать как из командной строки, так и с помощью встроенных средств автоматизации выполнения задач. Они делятся на две категории: команды для взаимодействия с виртуальными машинами в рамках эксперимента (табл. 2) и команды для изменения конфигурации виртуального экспериментального стенда (табл. 3).

Так как команды из табл. 3 меняют структуру эксперимента, их использование возможно только через CLI. Большинство команд из табл. 2 применимо как с помощью CLI для тестирования и отладки отдельных действий, так и с помощью встроенного средства автоматизации выполнения задач. Исключениями являются команды «vm status» и «vm ssh», которые применимы только через CLI.

[17] https://nodejs.org/, дата обращения: 18.02.2022. [https://nodejs.org/. Accessed February 18, 2022.]

[18] https://www.npmjs.com/, дата обращения: 18.02.2022. [https://www.npmjs.com/. Accessed February 18, 2022.]

[19] Скаффолдинг (англ. scaffolding) – метод программирования, предусматривающий генерацию программного кода для решения типовых задач, таких как, например, построение файловой структуры проекта или создание классов для доступа к таблицам базы данных. [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.]

Применение разработанного инструмента на практике предполагается по следующей методике:

1) планирование и дизайн экспериментального исследования;
2) инициализация проекта экспериментального исследования с заданным числом виртуальных машин;
3) подготовка формального описания виртуальных машин и их конфигураций в формате YAML;
4) разработка и отладка последовательности действий в эксперименте с помощью CLI-инструментария;
5) автоматизация последовательности действий в эксперименте с помощью инструментария автоматизации выполнения задач;
6) проведение эксперимента в программно-управляемом режиме;
7) анализ экспортированных данных об утилизации вычислительных ресурсов и иных данных, полученных в ходе эксперимента.

ЗАКЛЮЧЕНИЕ

В работе были проанализированы задачи, стоящие перед исследователями, и инструменты, с помощью которых осуществляется постановка экспериментов по оценке характеристик информационно-технологических решений. Выявлено, что проведение экспериментальных исследований требует от исследователя знаний и навыков работы с большим числом отдельных инструментов. Сформулированы три ключевые характеристики, которым должен соответствовать фреймворк.

Спроектирован предметно-ориентированный фреймворк, инкапсулирующий часть используемых технологий и предоставляющий необходимый функционал для проведения экспериментальных исследований. Подготовлена схема взаимодействия фреймворка с соответствующим ПО для проведения эксперимента. Определена общая схема проведения эксперимента на основе виртуальных машин. Также сформирована файловая структура проекта эксперимента, структура основных файлов проекта и внутренние связи между файлами проекта.

На основе результатов проектирования разработан фреймворк, содержащий 12 команд для работы с виртуальными машинами. Бóльшая часть команд возможна к исполнению как в режиме CLI, так и в режиме автоматизации выполнения задач. Для упрощения процесса подготовки проекта эксперимента реализованы 4 команды скаффолдинга. Также предложена методика использования фреймворка на практике, и описаны программные технологии, с помощью которых фреймворк реализован.

Таблица 2. Список команд взаимодействия с виртуальными машинами

Команда	Опции		Описание команды
	Имя	Описание	
vm compile	-n, --name	Имя виртуальной машины	Компилирует настройки виртуальных машин из слабоструктурированного формата данных в форматы конфигураций используемых инструментальных средств
	-s, --stage	Имя стадии эксперимента	
vm copy	-n, --name	Имя виртуальной машины	Копирует файл или директорию между хост-системой и указанной виртуальной машиной
	-s, --stage	Имя стадии эксперимента	
	-d, --direction	Copy direction	
	-f, --from	Copy from	
	-t, --to	Copy to	
vm destroy	-n, --name	Имя виртуальной машины	Удаляет все данные, связанные с виртуальной машиной (не конфигурации)
	-s, --stage	Имя стадии эксперимента	
vm exec	-n, --name	Имя виртуальной машины	Выполняет команду на указанной виртуальной машине
	-s, --stage	Имя стадии эксперимента	
	-c, --command	Команда для выполнения на виртуальной машине	
vm provision	-n, --name	Имя виртуальной машины	Производит установку и настройку ПО виртуальных машин в соответствии с конфигурациями провизии
	-s, --stage	Имя стадии эксперимента	
vm report	-n, --name	Имя виртуальной машины	Формирует отчет о задействовании вычислительных ресурсов виртуальными машинами. В скобках указан вариант в случае использования автоматизации
	-s, --stage	Имя стадии эксперимента	
	--start	Время начала периода для отчета (или название стадии эксперимента)	
	--end	Время завершения периода для отчета (или название стадии эксперимента)	
	-l, --labels	Список меток *Atop* для формирования отчета по ним	
vm restart	-n, --name	Имя виртуальной машины	Перезапускает виртуальную машину
	-s, --stage	Имя стадии эксперимента	
vm setupHosts	-n, --name	Имя виртуальной машины	Добавляет связки IP-Hostname каждой виртуальной машины в эксперименте в файл /etc/hosts виртуальных машин с целью упрощения адресации
	-s, --stage	Имя стадии эксперимента	
vm ssh	-n, --name	Имя виртуальной машины	Подключение к виртуальной машине посредством SSH
vm start	-n, --name	Имя виртуальной машины	Запускает виртуальную машину
	-s, --stage	Имя стадии эксперимента	
vm status	-n, --name	Имя виртуальной машины	Отображает статус виртуальной машины
	-s, --stage	Имя стадии эксперимента	
vm stop	-n, --name	Имя виртуальной машины	Выключает виртуальную машину
	-s, --stage	Имя стадии эксперимента	

Таблица 3. Список команд взаимодействия с проектом

Команда	Опции		Описание команды
	Имя	Описание	
project init	Отсутствуют, используется интерактивный режим ввода параметров		Инициализирует проект эксперимента в текущей директории и создает базовую файловую структуру
project run	-s, --stage	Имя стадии эксперимента	Запускает исполнение сконфигурированной последовательности действий эксперимента или указанной стадии эксперимента
project vm add	Отсутствуют, используется интерактивный режим ввода параметров		Добавляет новую конфигурацию виртуальной машины
project vm remove	-n, --name	Имя виртуальной машины	Удаляет существующую конфигурацию виртуальной машины

Дальнейшие исследования могут быть направлены на решение конкретных задач оценки информационно-технологических решений с применением фреймворка, выявление и устранение ограничений фреймворка, создание методической базы для проведения экспериментальных исследований с помощью предложенного инструментария.

БЛАГОДАРНОСТИ

Работа выполнена при финансовой поддержке гранта РТУ МИРЭА «Инновации в реализации приоритетных направлений развития науки и технологий», номер проекта НИЧ 28/22.

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.

СПИСОК ЛИТЕРАТУРЫ / 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

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

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