АВТОМАТИЗАЦИЯ И УПРАВЛЕНИЕ
AUTOMATION AND CONTROL

УДК 004.94 **DOI: 10.18413/2518-1092-2024-9-3-0-5**

Миронов Т.О.
Середенко Н.Н.

ПОСТРОЕНИЕ МОДЕЛИ РАСЧЕТА ВРЕМЕНИ ВЫПУСКА ДОРАБОТОК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПОСЛЕ ВНЕДРЕНИЯ СИСТЕМЫ УПРАВЛЕНИЯ РЕЛИЗНЫМ ЦИКЛОМ

Российский экономический университет им. Г.В. Плеханова, Стремянный переулок, 36, г. Москва, 117997, Россия

e-mail: tihon.mirovnoff@yandex.ru, Seredenko.NN@rea.ru

Аннотация
В работе предложена математическая модель по расчету времени выпуска нового релиза ПО. Данная модель позволяет оценить разницу до внедрения автоматизированной системы релизного цикла и после. Актуальность предложенной методики обусловлена возможностью ее применения к задаче оценки эффективности информационного комплекса, автоматизируемого процесс релизного цикла.
Объектом исследования является информационный департамент банка, предметом автоматизации – процесс оценки времени выпуска доработок программного обеспечения.
Целью работы является построение модели расчета времени выпуска программных доработок после внедрения системы управления релизным циклом, обеспечивающей регулярное, быстрое и стабильное внедрение новых версий программного обеспечения. Для достижения поставленной цели в рамках работы спроектировано архитектурное решение системы, автоматизирующей процесс релизного цикла; смоделирован новый процесс взаимодействия персонала после внедрения системы релизного цикла; разработан конвейер автоматической доставки ПО в промышленную среду и процесс автоматической упаковки ПО в контейнеры; построена математическая модель расчета времени выпуска доработок программного обеспечения после системы управления релизным циклом.
Для решения перечисленных задач использовалась система Gitlab класса CI/CD, предоставляющая возможность автоматически собирать контейнеры и деплоить их в окружения, и система Deckhouse для оркестровки контейнеров.
Ключевые слова: гибкие методологии разработки; DevOps; Infrastructure as Code (IaC); автоматический релизный цикл; воспроизводимость программного обеспечения; расчет времени выпуска ПО; управление релизным циклом ПО
Для цитирования: Миронов Т.О., Середенко Н.Н. Построение модели расчета времени выпуска доработок программного обеспечения после внедрения системы управления релизным циклом // Научный результат. Информационные технологии. – Т.9, №3, 2024. – С. 43-54. DOI: 10.18413/2518-1092-2024-9-3-0-5

Mironov T.O.
Seredenko N.N.

BUILDING A MODEL FOR CALCULATING THE RELEASE TIME OF SOFTWARE IMPROVEMENTS AFTER THE IMPLEMENTATION OF THE RELEASE CYCLE MANAGEMENT SYSTEM

Plekhanov Russian University of Economics,
36 Stremyanny lane, Moscow, 117997, Russia

e-mail: tihon.mirovnoff@yandex.ru, Seredenko.NN@rea.ru

Abstract
The paper proposes a mathematical model for calculating the release time of a new software release. This model allows you to evaluate the difference between the operation of a production system

during the release cycle and after it. The relevance of the proposed methodology determines the need for its application to the problem of assessing the effectiveness of an information complex and an automated release process cycle.
The object of the study is the information department of the bank, ensuring automation – the process of estimating the release time of software improvements.
The goal of the work is to build a model for calculating the release time of software improvements after the implementation of a release cycle management system, ensuring regular, fast and stable implementation of new software versions. To achieve this goal, within the framework of the work, an architectural solution for a system that automates the release cycle process was designed; a new process of personnel interaction was modeled after the implementation of the release cycle system; a pipeline for automatic delivery of software to an industrial environment and a process for automatic packaging of software into containers have been developed; a mathematical model was built to calculate the release time of software improvements after the release cycle management system.
To solve these problems, we used the Gitlab CI/CD class system, which provides the ability to automatically assemble containers and deploy them into environments, and the Deckhouse system for container orchestration.
Keywords: agile development methodologies; DevOps; Infrastructure as Code (IaC); automatic release cycle; software reproducibility; software release timing; software release cycle management
For citation: Mironov T.O., Seredenko N.N. Building a model for calculating the release time of software improvements after the implementation of the release cycle management system // Research result. Information technologies. – T. 9, №3, 2024. – P. 43-54. DOI: 10.18413/2518-1092-2024-9-3-0-5

ВВЕДЕНИЕ

В новейшем времени программные продукты стали многократно сложнее. Если, по каким-то причинам, еще нет программного продукта, закрывающего какую-либо нишу бизнеса, начинается гонка: кто быстрее выпустит продукт. Это, в свою очередь, привело к развитию гибких методологий разработки: теперь программное обеспечение оказывается на рынке как минимально жизненный продукт, вместе с «обещаниями» разработчиков и в последующем его развивать [2].

Для того, чтобы обеспечивать потребности рынка в быстром, непрерывном и стабильном выпуске обновлений программного обеспечения, архитектурный комплекс должен удовлетворять следующим требованиям:

- Создание независимых от среды сборки образов программного обеспечения. Это позволяет обеспечить воспроизводимость образов программного обеспечения независимо от среды разработки или развертывания. Достижение воспроизводимости позволяет устранить потенциальные проблемы совместимости [2], [3].
- Автоматическое развертывание в тестовой среде. Автоматизация процесса развертывания позволяет быстро и эффективно протестировать программное обеспечение в контролируемой среде, что способствует быстрому получению обратной связи по изменениям кода и сокращению времени, необходимого для тестирования [4].
- Автоматическое тестирование программного обеспечения. Автоматизация процедур тестирования позволяет быстро обнаружить дефекты и значительно сократить время цикла тестирования [5].
- Контролируемый деплой в продуктовой среде. В системе должны быть предусмотрены такие функции, как контроль версий, откат выпуска, что позволяет осуществлять плавное развертывание при сохранении высокой доступности системы.

Для того, чтобы оценить качество системы, обеспечивающей исполнение перечисленных функций, необходимо иметь критерии оценки информационного комплекса [1]. Одним из ключевых критериев оценки эффективности автоматизации релизного цикла является время выпуска нового

релиза. В данной работе предложена методика, позволяющая оценивать информационный комплекс по данному критерию.

ПОСТРОЕНИЕ АРХИТЕКТУРЫ СИСТЕМЫ УПРАВЛЕНИЯ РЕЛИЗНЫМ ЦИКЛОМ

В рамках анализа распространенных архитектурных решений по управлению релизным циклом ПО сформулирован перечень ключевых проблем. Предложения по их устранению и преимущества предлагаемых решений приведены в табл.1.

Таблица 1

Перечень ключевых проблем типовой архитектуры (составлено автором)

Table 1

List of key problems of a typical architecture (compiled by the author)

№	Проблема	Предложения по устранению проблемы	Преимущества предлагаемого решения
1	Разработчикам приходится тратить свое время на настройку окружения для нового выпуска ПО: подключаться на предоставленные им сервера, переносить код из репозитория, собирать его, а также устанавливать зависимости [14]	Более тесное взаимодействие команд разработки и администрирования. Создание единого pipeline для сборки	Сокращение издержек для команды разработки. Оперативное отслеживание изменений для команды администраторов
2	При конфигурировании промышленной среды может обнаружится, что настройки для работоспособности релиза из тестовых сред не были задокументированы [12], [13]	Использование IaC (инфраструктура как код), единого декларативного описания желаемого состояния системы	Централизованное управление и изменение серверных конфигураций, получение одинакового результата на всех системах
3	Различие тестовых и промышленных серверных окружений, что создает сложности администрирования и сборку программных образов под каждую архитектуру [16]	Использование технологий контейнеризации [17]	Сокращение количества сборочных пакетов, ускоренная доставка образов
4	Сложность управления взаимосвязанными компонентами ПО [15]	Использование технологий контейнерной оркестрации	Описание взаимосвязей компонентов как IaC. Управление развертыванием системы через конфигурацию

Представленные в таблице 1 недостатки дополнительно обуславливают переход к гибким методологиям проектирования.

Также распространенной проблемой популярных решений по организации систем управления релизным циклом является тот факт, что в зависимости от среды исполнения код может вести себя по-разному [20]. Для решения этого недостатка предлагается воспользоваться технологиями контейнеризации, которые упаковывают код со всеми его зависимостями и изолируют его от среды выполнения. Для управления группами контейнеров предназначены системы оркестровки контейнеров.

В рамках работы был составлен перечень критериев для оценки систем оркестровки контейнеров. По полученным критериям произведена оценка систем на соответствие, результат представлен в таблице 2.

Таблица 2

Сравнительный анализ систем оркестровки контейнеров (составлено автором)

Table 2

Comparative analysis of container orchestration systems (compiled by the author)

№	Критерий	Docker Swarm	Openshift	Deckhouse	K8s [19]
1.	Простота установки	Соответствует	Частично соответствует	Соответствует	Полностью не соответствует
2.	Интуитивно понятный пользовательский интерфейс	Частично соответствует	Частично соответствует	Соответствует	Полностью не соответствует
3.	Простота обслуживания	Частично соответствует	Соответствует	Частично соответствует	Частично соответствует
4.	Bare metal. Возможность установки напрямую на аппаратную платформу	Соответствует	Соответствует	Соответствует	Соответствует
5.	Автоматическая балансировка нагрузки	Полностью не соответствует	Соответствует	Соответствует	Соответствует
6.	Возможность настройки централизованного аудита в сторонние системы	Частично соответствует	Соответствует	Соответствует	Частично соответствует
7.	Автоматическое обновление ПО в кластере и наличие управления версиями	Полностью не соответствует	Соответствует	Соответствует	Соответствует
8.	Поддержка российских ОС	Полностью не соответствует	Полностью не соответствует	Соответствует	Полностью не соответствует
9.	Интеграция с решениями CI/CD	Частично соответствует	Соответствует	Соответствует	Соответствует

№	Criteria	Docker Swarm	Openshift	Deckhouse	K8s [19]
1.	Easy to install	Corresponds	Частично соответствует	Corresponds	Does not correspond
2.	Интуитивно понятный пользовательский интерфейс	Частично соответствует	Частично соответствует	Corresponds	Does not correspond
3.	Easy maintenance	Partially corresponds	Corresponds	Partially corresponds	Partially corresponds
4.	Bare metal. The ability to install directly on the hardware platform.	Corresponds	Corresponds	Corresponds	Corresponds
5.	Automatic load balancing	Does not correspond	Corresponds	Corresponds	Corresponds
6.	The ability to configure centralized auditing in third-party systems	Partially corresponds	Corresponds	Corresponds	Partially corresponds
7.	Automatic software updates in the cluster and the availability of version control	Does not correspond	Corresponds	Corresponds	Corresponds
8.	Support for Russian operating systems	Does not correspond	Does not correspond	Corresponds	Does not correspond
9.	Integration with CI/CD solutions	Partially corresponds	Corresponds	Corresponds	Corresponds

На основании сравнения систем, проведённого в таблице 2, можно сделать вывод о том, что Deckhouse является оптимальным по выделенным критериям.

Также для решения поставленных задач необходима система класса CI/CD, предоставляющая возможность автоматически собирать контейнеры и деплоить их в окружения.

В рамках работы составлен перечень критериев для оценки систем CI/CD. По полученным критериям была произведена оценка систем на соответствие, результаты представлены в таблице 3.

Таблица 3

Сравнительный анализ систем CI/CD (составлено автором)

Table 3

Comparative analysis of CI/CD systems (compiled by the author)

№	Критерий	Gitlab	Jenkins	TeamCity	Bamboo
1.	Простота установки	Соответствует	Частично соответствует	Соответствует	Соответствует
2.	Интуитивно понятный пользовательский интерфейс	Соответствует	Частично соответствует	Соответствует	Полностью не соответствует
3.	Простота обслуживания	Частично соответствует	Не соответствует	Частично соответствует	Соответствует
4.	Автоматическая балансировка нагрузки	Частично соответствует	Частично соответствует	Частично соответствует	Частично соответствует
5.	Возможность настройки централизованного аудита в сторонние системы	Соответствует	Соответствует	Соответствует	Частично соответствует
6.	Поддержка российских ОС	Полностью не соответствует	Полностью не соответствует	Соответствует	Полностью не соответствует
7.	Интеграция с решениями версионирования	Соответствует	Частично соответствует	Частично соответствует	Частично соответствует

По результатам сравнительного анализа для хранения кода и автоматизации CI/CD пайплайна в работе выбран программный продукт GitLab.

Таким образом, концептуальный дизайн системы представляет собой виртуальные машины, которые развернуты в кластере гипервизора (рис. 1). Для хранения кода и автоматизации CI/CD пайплайна используется программный продукт GitLab, для организации системы оркестрации контейнеров используется система Deckhouse [18].

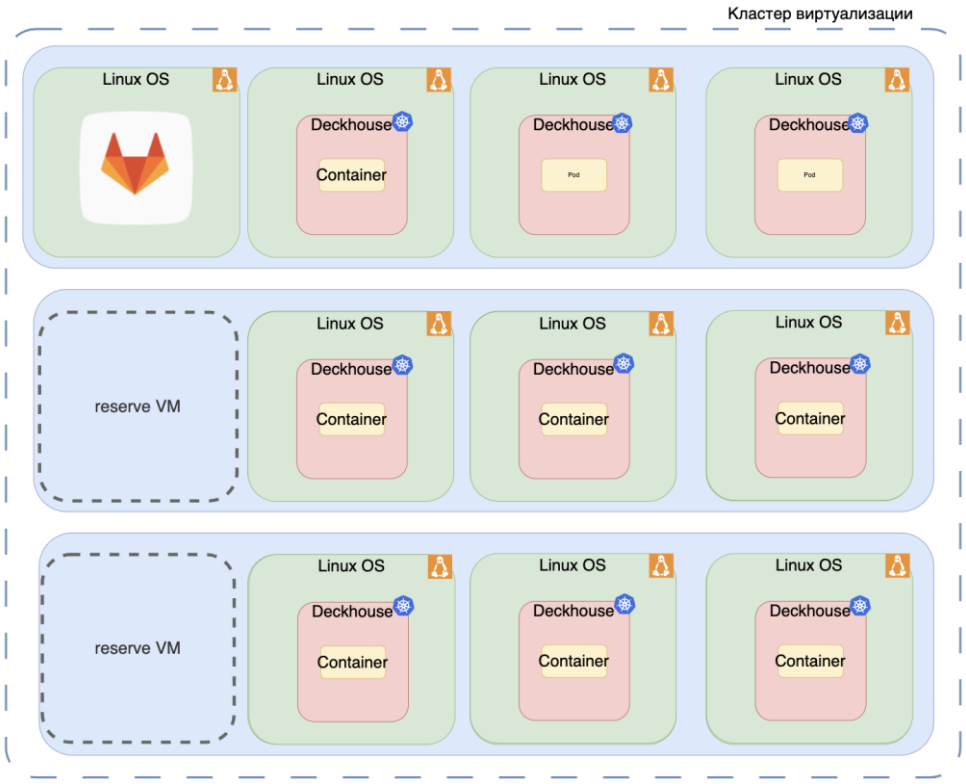

Рис. 1. Концептуальный дизайн системы (составлено автором)
Fig. 1. Conceptual design of the system (compiled by the author)

Схема работы Deckhouse представлена на рисунке 2: система состоит из 3 master, обеспечивающих центральное управление кластером, и 6 worker нод, позволяющих запускать на себе полезную нагрузку. Таким образом, поды с полезной нагрузкой представляют собой контейнеры с разрабатываемым ПО.

Программный код информационного комплекса предприятия обычно разбивается на проекты, которые представляют собой набор репозиториев. Организация репозиториев производится по крупным смысловым блокам, которые обновляются при внесении программных доработок. История их версий содержится в соответствующей сущности репозитории. Обновление репозиториев происходит путем включения веток с изменениями. Таким образом, появляются запросы на слияние, которые регистрирует программист при завершении изменений в ветке [2], [6], [11].

Конвейер представляет собой набор необходимых задач на выполнения для раннера. Администратор системы разрабатывает автоматические инструкции для работы конвейера и далее конвейер выполняет задачи автоматически в соответствии с инструкциями [8], [9].

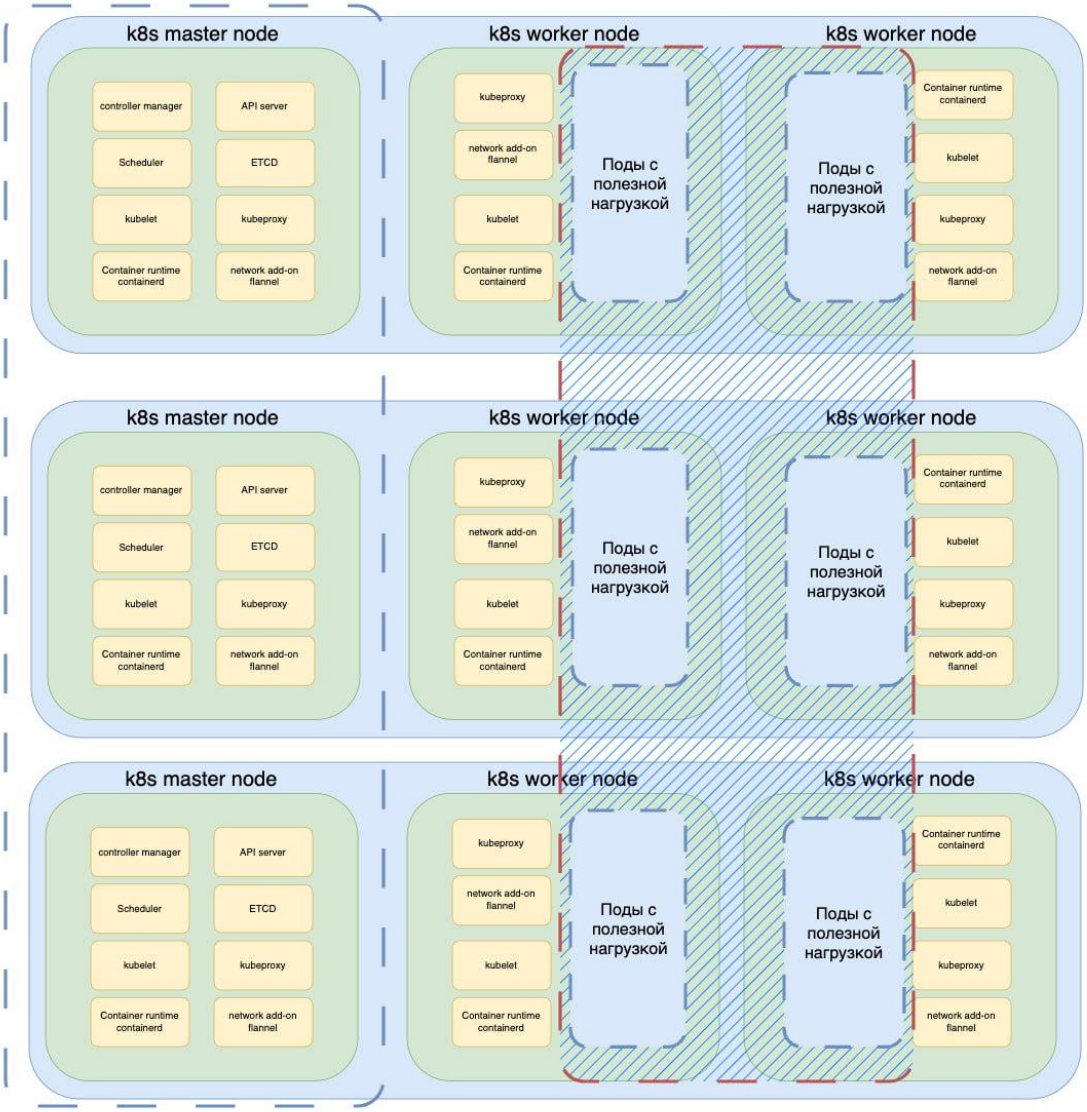

Рис. 2. Схема работы Deckhouse (составлено автором)
Fig. 2. The scheme of Deckhouse operation (compiled by the author)

РАСЧЕТ ВРЕМЕНИ ВЫПУСКА ДОРАБОТОК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

На основе анализа внедрения реальных проектов предложим методику расчета, позволяющую оценить влияние программно-аппаратных решений, обеспечивающих автоматизацию цикла выпуска программного обеспечения, на итоговый расчетный показатель скорости разработки продукта.

Время выпуска новой функциональности программного обеспечения можно рассчитать по формуле (1):

$$Rt = DEVt + Tt + DEPt + c, \quad (1)$$

где Rt – время выпуска нового релиза, $DEVt$ – время разработки нового функционала; Tt – время тестирования нового функционала, $DEPt$ – время развертывания нового релиза в среде эксплуатации, с – человеческий фактор.

В свою очередь, время тестирования можно представить как формулу (2):

$$Tt = n \times (DEPt + TSt + Ft), \quad (2)$$

где $DEPt$ – время развертывания нового релиза в среде эксплуатации, TSt – время выполнения тестов: интеграционных, системных и др., Ft – время устранения дефектов, n – количество итераций тестирования.

Время настройки среды можно представить как формулу (3):

$$DEPt = SERVt + ICt + INSt, \quad (3)$$

где $SERVt$ – время подготовки инфраструктуры, ICt – время упаковки кода в образы, $INSt$ – время инсталляции образов в среде эксплуатации.

Подставим выражение для расчета времени настройки среды и получим формулу (4):

$$Rt = DEVt + n \times (SERVt + ICt + INSt + TSt + Ft) + SERVt + ICt + INSt, \quad (4)$$

Так как величины зависят от внешних факторов (например: скорость серверной инфраструктуры), далее будет использоваться нотация Big O [21], что позволит определить зависимость количества операций от входных параметров, без привязки ко времени.

Уберем из формулы переменные, которые не зависят от системы и получим формулу (5):

$$Rt = n \times (SERVt + ICt + INSt) + SERVt + ICt + INSt, \quad (5)$$

Сократим, получится формула (6):

$$Rt = (n + 1) \times (SERVt + ICt + INSt), \quad (6)$$

Что в свою очередь, является формулой (7):

$$Rt = n \times (SERVt + ICt + INSt), \quad (7)$$

Что в нотации Big O можно представить как формулу (8):

$$Rt = O(n), \quad (8)$$

Т.е. получаем линейную зависимость времени выпуска нового релиза от количества итераций тестирования, что позволяет построить график (рисунок 3).

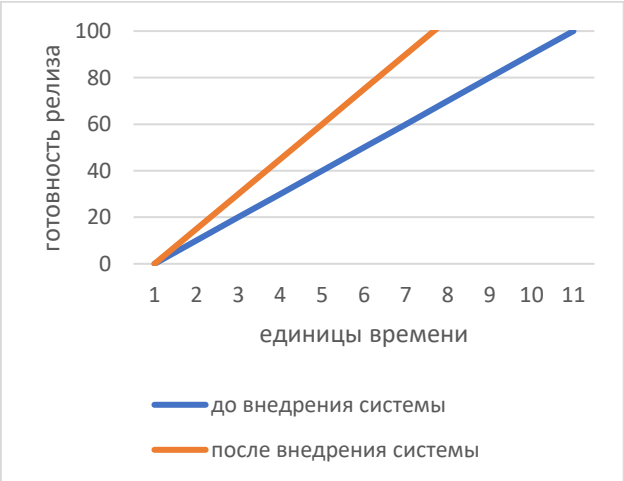

Рис. 3. График скорости разработки продукта до и после внедрения системы. (составлено автором)
Fig. 3. Product development speed chart before and after system installation. (compiled by the author)

Для сравнения скорости релиза используя формулу линейной функции (9):

$$y = kx + b, \quad (9)$$

где y – готовность релиза, x – время разработки релиза, k – коэффициент скорости разработки, b – смещение начала разработки релиза (для наглядности $b = 0$).

ЗАКЛЮЧЕНИЕ

Представленная математическая модель позволяет оценивать время выпуска релиза и проводить сравнительный анализ архитектурных решений, автоматизирующих процесс релизного цикла, а значит представляет интерес для организаций, занимающихся развитием и доработкой информационных систем [6].

В работе представлена математическая модель, позволяющая оценивать подход к разработке архитектурной модели программно-аналитического комплекса, обеспечивающего автоматизацию цикла выпуска программного обеспечения. Предложенная архитектура является обобщением, а в реальной практике могут встречаться отклонения от представленной архитектуры. Данные отклонения обусловлены различным исходным уровнем информатизации предприятий, финансовыми ресурсами, которыми располагает предприятие, и прочими факторами, являющимися индивидуальными для каждой конкретной организации.

В качестве направлений для дальнейших исследований можно выделить следующие:

- Апробация модели в реальном секторе.
- Анализ применимости предложенной архитектуры на предприятиях, дорабатывающих системы разного типа: монолитные информационные системы, микро-сервисные комплексы и т.п.
- Применение математической модели расчётов в реальных проектах, оценка влияния первоначального уровня автоматизации процесса на итоговый расчетный показатель скорости разработки продукта.

Список литературы

1. Баранов С.Н. Метрическое обеспечение программных разработок / С.Н. Баранов, А.М. Тележкин // Труды СПИИРАН. – 2014. – № 5(36). – С. 5-27. – EDN TELOAV.
2. Голицына О. Л. Информационные системы: учебное пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. – 2-е изд. – Москва: ФОРУМ: ИНФРА-М, 2022. – 448 с.: ил. – (Высшее образование).
3. Григорьев Д.Ю. Автоматизация разработки программного обеспечения при помощи методологии CI/CD / Д.Ю. Григорьев, Н.В. Гайдук // Информационное общество: современное состояние и перспективы развития: Сборник материалов XIV международного форума, Краснодар, 12–17 июля 2021 года. – Краснодар: Кубанский государственный аграрный университет имени И.Т. Трубилина, 2021. – С. 110-113. – EDN LWXQPX.
4. Игорихина Е.В. Гибкие методологии разработки программного обеспечения / Е.В. Игорихина, О.В. Михайлова // Ломоносовские чтения на Алтае: фундаментальные проблемы науки и образования: избранные труды международной конференции, Барнаул, 14–17 ноября 2017 года / Алтайский государственный университет. Том Часть 1. – Барнаул: Алтайский государственный университет, 2017. – С. 232-237. – EDN YPTRBL.
5. Quattrocchi G. Infrastructure as Code / G. Quattrocchi, D.A. Tamburri // IEEE Software. – 2023. – Vol. 40, No. 1. – P. 37-40. – DOI 10.1109/ms.2022.3212034. – EDN YRDJTS.
6. Осипова Е.Е. Проблемы внедрения гибких методологий разработки в ИТ-компанию / Е.Е. Осипова, В.В. Жуков // Естествознание и технические науки: глобальные вызовы, тренды, возможности: сборник научных трудов по материалам Международной научно-практической конференции, Белгород, 30 мая 2019 года / Агентство перспективных научных исследований (АПНИ). – Белгород: Общество с ограниченной ответственностью "Агентство перспективных научных исследований", 2019. – С. 90-93. – EDN ICXVFD.
7. Artac M., Borovssak T., Di Nitto E., Guerriero M., Tamburri D. ‘DevOps: Introducing Infrastructure-As-Code’. 2017 IEEE/ACM 39Th International Conference On Software Engineering Companion (ICSE-C).
8. Херинг М. DevOps для современного предприятия: методическое пособие / М. Херинг; пер. с анг. М. А. Райтмана. – Москва: ДМК Пресс, 2020. – 232 с.
9. Forsgren N. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations / IT Revolution Press; 1st edition (March 27, 2018) – 288 p.

10. Rahman A., Parnin C., Williams L. 'The Seven Sins: Security Smells in Infrastructure as Code Scripts'. IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019.

11. Jóźwiak I.J. Current infrastructure as a code automation trends in context of cloud agnostic resource provisioning / I.J. Jóźwiak, P.P. Jóźwiak, K. Zatwarnicki // Scientific Papers of Silesian University of Technology Organization and Management Series. – 2023. – Vol. 2023, No. 186. – P. 167-183. – DOI 10.29119/1641-3466.2023.186.13. – EDN TLMVAE.

12. Infrastructure as Code for Security Automation and Network Infrastructure Monitoring / W.R.A. Putra, A.R.A. Nurwa, D.F. Priambodo, M. Hasbi // MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer. – 2022. – Vol. 22, No. 1. – P. 201-214. – DOI 10.30812/matrik.v22i1.2471. – EDN UXLFWV.

13. Косенков В.В. Сравнительный анализ инфраструктурных инструментов Infrastructure as Code / В.В. Косенков, А.В. Елфимов // Современные стратегии и цифровые трансформации устойчивого развития общества, образования и науки: Сборник материалов VI Международной научно-практической конференции, Москва, 10 февраля 2023 года. – Москва: Общество с ограниченной ответственностью "Издательство АЛЕФ", 2023. – С. 81-86. – EDN CSIPEK.

14. Integration of Security Standards in DevOps Pipelines: An Industry Case Study / F. Moyón, R. Soares, M. Pinto-Albuquerque [et al.] // Lecture Notes in Computer Science. – 2020. – Vol. 12562 LNCS. – P. 434-452. – DOI 10.1007/978-3-030-64148-1_27. – EDN GQTKYQ.

15. Deployment and communication patterns in microservice architectures: A systematic literature review / I. Karabey Aksakalli, T. Çelik, A.B. Can, B. Teki Nerdoğan // Journal of Systems and Software. – 2021. – Vol. 180. – P. 111014. – DOI 10.1016/j.jss.2021.111014. – EDN XKZDNS.

16. Multi-objective microservice deployment optimization via a knowledge-driven evolutionary algorithm / W. Ma, R. Wang, Yu. Gu [et al.] // Complex and Intelligent Systems. – 2021. – Vol. 7, No. 3. – P. 1153-1171. – DOI 10.1007/s40747-020-00180-1. – EDN CLUAOB.

17. A container deployment strategy for server clusters with different resource types / M. Ouyang, J. Xi, W. Bai, K. Li // Concurrency Computation Practice and Experience. – 2023. – Vol. 35, No. 10. – DOI 10.1002/cpe.7665. – EDN THCQSF.

18. Свидетельство о государственной регистрации программы для ЭВМ № 2022661210 Российская Федерация. Deckhouse Kubernetes Platform: № 2022612892: заявл. 04.03.2022: опубл. 17.06.2022 / Д.О. Столяров, П.С. Головин, А.Г. Кладов [и др.]; заявитель Акционерное общество "Флант". – EDN JPVZDR.

19. Сальков А.А. Сравнение Kubernetes-платформы Deckhouse с Vanilla Kubernetes / А.А. Сальков, Н.Н. Лытнев, А.М. Кумратова // Информационное общество: современное состояние и перспективы развития: Сборник материалов XV международного форума, Краснодар, 10–14 июля 2023 года. – Краснодар: Кубанский государственный аграрный университет имени И.Т. Трубилина, 2023. – С. 268-270. – EDN ITYFMR.

20. Сергеева А.С. Управление релизами при доработке IT-продуктов / А.С. Сергеева, Л.Г. Ахметшина // Самоуправление. – 2020. – № 4(121). – С. 458-461. – EDN NKDVGW.

21. Big O notation: [сайт]. URL: https://en.wikipedia.org/wiki/Big_O_notation

References

1. Baranov S.N. Metric support of software developments / S.N. Baranov, A.M. Telezhkin // Proceedings of SPIIRAS. – 2014. – No. 5 (36). – P. 5-27. – EDN TELOAV.

2. Golitsyna O.L. Information systems: a tutorial / O.L. Golitsyna, N.V. Maksimov, I.I. Popov. – 2nd ed. – Moscow: FORUM: INFRA-M, 2022. – 448 p.: ill. – (Higher education).

3. Grigoriev D.Yu. Automation of software development using the CI/CD methodology / D.Yu. Grigoriev, N.V. Gaiduk // Information society: current state and development prospects: Collection of materials of the XIV international forum, Krasnodar, July 12-17, 2021. – Krasnodar: Kuban State Agrarian University named after I.T. Trubilin, 2021. – P. 110-113. – EDN LWXQPX.

4. Igorikhina E.V. Flexible methodologies for software development / E.V. Igorikhina, O.V. Mikhailova // Lomonosov readings in Altai: fundamental problems of science and education: selected proceedings of the international conference, Barnaul, November 14-17, 2017 / Altai State University. Volume Part 1. – Barnaul: Altai State University, 2017. – P. 232-237. – EDN YPTRBL.

5. Quattrocchi G. Infrastructure as Code / G. Quattrocchi, D.A. Tamburri // IEEE Software. – 2023. – Vol. 40, No. 1. – P. 37-40. – DOI 10.1109/ms.2022.3212034. – EDN YRDJTS.

6. Osipova E.E. Problems of implementing flexible development methodologies in an IT company / E.E. Osipova, V.V. Zhukov // Natural Science and Engineering: Global Challenges, Trends, Opportunities: Collection of Scientific Papers Based on the Materials of the International Scientific and Practical Conference,

Belgorod, May 30, 2019 / Agency for Advanced Scientific Research (APNI). – Belgorod: Limited Liability Company "Agency for Advanced Scientific Research", 2019. – P. 90-93. – EDN ICXVFD.

7. Artac M., Borovssak T., Di Nitto E., Guerriero M. Tamburri, D. 'DevOps: Introducing Infrastructure-As-Code'. 2017 IEEE/ACM 39Th International Conference On Software Engineering Companion (ICSE-C).

8. Hering M. DevOps for the Modern Enterprise: A Methodological Handbook / M. Hering; trans. from English by M. A. Reitman. – Moscow: DMK Press, 2020. – 232 p.

9. Forsgren N. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations / IT Revolution Press; 1st edition (March 27, 2018) – 288 pages.

10. Rahman A., Parnin C., Williams L. 'The Seven Sins: Security Smells in Infrastructure as Code Scripts'. IEEE/ACM 41st International Conference on Software Engineering (ICSE), 2019.

11. Jóźwiak I.J. Current infrastructure as a code automation trends in the context of cloud agnostic resource provisioning / I.J. Jóźwiak, P.P. Jóźwiak, K. Zatwarnicki // Scientific Papers of Silesian University of Technology Organization and Management Series. – 2023. – Vol. 2023, No. 186. – P. 167-183. – DOI 10.29119/1641-3466.2023.186.13. – EDN TLMVAE.

12. Infrastructure as Code for Security Automation and Network Infrastructure Monitoring / W.R.A. Putra, A.R.A. Nurwa, D.F. Priambodo, M. Hasbi // MATRIK: Jurnal Management, Teknik Informatika dan Rekayasa Komputer. – 2022. – Vol. 22, No. 1. – P. 201-214. – DOI 10.30812/matrik.v22i1.2471. – EDN UXLFWV.

13. Kosenkov V.V. Comparative analysis of Infrastructure as Code infrastructure tools / V.V. Kosenkov, A.V. Elfimov // Modern strategies and digital transformations of sustainable development of society, education and science: Collection of materials of the VI International scientific and practical conference, Moscow, February 10, 2023. – Moscow: Limited Liability Company "Izdatelstvo ALEF", 2023. – P. 81-86. – EDN CSIPEK.

14. Integration of Security Standards in DevOps Pipelines: An Industry Case Study / F. Moyón, R. Soares, M. Pinto-Albuquerque [et al.] // Lecture Notes in Computer Science. – 2020. – Vol. 12562 LNCS. – P. 434-452. – DOI 10.1007/978-3-030-64148-1_27. – EDN GQTKYQ.

15. Deployment and communication patterns in microservice architectures: A systematic literature review / I. Karabey Aksakalli, T. Çelik, A.B. Can, B. Teki Nerdoğan // Journal of Systems and Software. – 2021. – Vol. 180. – P. 111014. – DOI 10.1016/j.jss.2021.111014. – EDN XKZDNS.

16. Multi-objective microservice deployment optimization via a knowledge-driven evolutionary algorithm / W. Ma, R. Wang, Yu. Gu [et al.] // Complex and Intelligent Systems. – 2021. – Vol. 7, No. 3. – P. 1153-1171. – DOI 10.1007/s40747-020-00180-1. – EDN CLUAOB.

17. A container deployment strategy for server clusters with different resource types / M. Ouyang, J. Xi, W. Bai, K. Li // Concurrency Computation Practice and Experience. – 2023. – Vol. 35, No. 10. – DOI 10.1002/cpe.7665. – EDN THCQSF.

18. Certificate of state registration of computer program No. 2022661210 Russian Federation. Deckhouse Kubernetes Platform: No. 2022612892: declared 03/04/2022: published 06/17/2022 / D.O. Stolyarov, P.S. Golovin, A.G. Kladov [et al.]; applicant Joint-Stock Company "Flant". – EDN JPVZDR.

19. Salkov A.A. Comparison of the Deckhouse Kubernetes Platform with Vanilla Kubernetes / A.A. Salkov, N.N. Lytnev, A.M. Kumratova // Information Society: Current State and Development Prospects: Collection of Materials of the XV International Forum, Krasnodar, July 10-14, 2023. – Krasnodar: Kuban State Agrarian University named after I. T. Trubilin, 2023. – Pp. 268-270. – EDN ITYFMR.

20. Sergeeva A.S. Release Management in the Finalization of IT Products / A.S. Sergeeva, L.G. Akhmetshina // Self-Government. – 2020. – No. 4 (121). – Pp. 458-461. – EDN NKDVGW.

21. Big O notation: [website]. URL: https://en.wikipedia.org/wiki/Big_O_notation

Миронов Тихон Олегович, студент 4-го курса факультета "Плехановская школа бизнеса "Интеграл"
Середенко Наталья Николаевна, кандидат экономических наук, доцент кафедры Прикладной информатики и информационной безопасности

Mironov Tikhon Olegovich, 4th year student of the faculty "Plekhanov School of Business "Integral"
Seredenko Natalya Nikolaevna, Candidate of Economic Sciences, Associate Professor of the Department of Applied Informatics and Information Security