УДК 004.415.2

Еременкова Валерия Вячеславовна, магистрант, Национальный исследовательский технологический университет, г. Москва

Ципес Григорий Львович, к.э.н., доцент кафедры «Магистерская школа информационных бизнес-систем», г. Москва

УПРАВЛЕНИЕ ИЗМЕНЕНИЯМИ В КОРПОРАТИВНЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ НА БАЗЕ LOW-CODE ПЛАТФОРМЫ KNOWLEDGE SPACE

Аннотация.

В статье рассматривается подход к управлению изменениями конфигураций корпоративных информационных систем на базе low-code платформы Knowledge Space с использованием двухсредовой модели dev–prod. Показано, что разделение среды разработки и промышленной среды позволяет снизить риски несанкционированных изменений, повысить воспроизводимость конфигураций и обеспечить контроль жизненного цикла решений. На основе анализа платформы Knowledge Space, современных публикаций по low-code и DevOps предложена укрупнённая схема жизненного цикла изменений, а также подход к интеграции платформы с Git-сервисом и GitLab. Практическая значимость работы состоит в возможности применения предложенного подхода для сопровождения корпоративных решений с повышенными требованиями к стабильности и управляемости.

Ключевые слова: low-code, Knowledge Space, dev, prod, управление изменениями, Git, GitLab, DevOps, конфигурация.

Annotation.

The article considers an approach to configuration change management in corporate information systems based on the Knowledge Space low-code platform using a two-environment dev–prod model. It is shown that separating the development and production environments reduces the risks of unauthorized changes, improves configuration reproducibility, and supports lifecycle control of enterprise solutions. Based on the analysis of the Knowledge Space platform and recent publications on low-code and DevOps, a generalized change lifecycle scheme and an approach to integrating the platform with a Git service and GitLab are proposed. The practical significance of the study lies in the possibility of applying the proposed approach to maintain corporate solutions with high requirements for stability and manageability.

Keywords: low-code, Knowledge Space, dev, prod, change management, Git, GitLab, DevOps, configuration.

ВВЕДЕНИЕ

Low-code платформы стали важным инструментом цифровой трансформации, так как они активно используются при необходимости ускорить создание корпоративных информационных систем ввиду визуального моделирования, повторного использования компонентов и уменьшения доли ручного программирования. По данным современных исследований, именно ускорение разработки и снижение зависимости от дефицитных ИТ-ресурсов объясняют рост интереса организаций к low-code решениям [1; 2]. Вместе с тем ускорение разработки повышает требования к управлению изменениями, особенно когда конфигурация прикладной системы модифицируется часто.

Для корпоративных платформ ключевой задачей становится обеспечение контролируемого переноса конфигурации из среды разработки в промышленную среду. В противном случае изменения могут вноситься напрямую в рабочую систему, что ухудшает воспроизводимость, усложняет анализ ошибок и

повышает риск отказов. В связи с этим для low-code среды требуется применение подходов, характерных для классической программной инженерии: разделение сред, контроль версий, резервное копирование и регламент выпуска изменений.

Цель статьи — представить компактную модель управления изменениями на платформе Knowledge Space, основанную на разделении сред dev и prod, а также показать, каким образом эта модель может быть усилена созданием Git-сервиса и интеграцией с GitLab.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ УПРАВЛЕНИЯ ИЗМЕНЕНИЯМИ В LOW-CODE СРЕДЕ

Исследования последних лет показывают, что low-code и no-code решения следует рассматривать не только как инструмент ускоренной разработки, но и как самостоятельную программную среду, к которой предъявляются требования качества, надёжности и сопровождаемости [1; 3; 4]. В отличие от классических подходов программирования, объектом изменений здесь являются не только программные модули, но и метаданные: формы, модели данных, справочники, правила, связи и вычисляемые показатели. Поэтому модификация конфигурации такой системы требует не менее жёсткого контроля, чем правка исходного кода.

С точки зрения управления жизненным циклом существенными являются два базовых положения. Первое связано с чётким разграничением разработческой и эксплуатационной частей. Второе предполагает обязательную фиксацию состояний конфигурации в системе версионирования или ином специализированном хранилище. Для низкокодовых платформ эта задача усложняется тем, что значительная часть логики размещается во внутренних структурах среды, однако современные работы и практический опыт показывают, что экспорт метаданных в текстовые форматы позволяет применять Git и DevOps-подходы и к этому классу решений [4; 6].

В применении к корпоративным информационным системам данная проблема приобретает особую остроту, поскольку ошибки в конфигурации приводят к искажению данных, нарушению маршрутов бизнес-процессов и остановке пользовательских операций. В результате управляемость изменений выступает не вспомогательной возможностью, а необходимым условием устойчивой эксплуатации.

2. ОСОБЕННОСТИ ПЛАТФОРМЫ KNOWLEDGE SPACE

Knowledge Space представляет собой корпоративную низкокодовую платформу, предназначенную для разработки прикладных решений, ориентированных на работу с сущностями, показателями, формулами, связями и объектами метамодели. С архитектурной точки зрения она объединяет пользовательский интерфейс, серверные сервисы, базы данных и прикладные модули, обеспечивающие хранение и обработку конфигурации.

В рамках рассматриваемой тематики ключевыми являются две характеристики этой среды. Во-первых, реализовано разделение на несколько контуров, среди которых для задач управления изменениями определяющими являются dev и prod. Во-вторых, предусмотрены встроенные механизмы хранения и резервного копирования конфигурации, которые при необходимости могут дополняться внешней системой контроля версий и отдельным Git-сервисом. Схема архитектуры платформы Knowledge Space проиллюстрирована на рисунке 1.

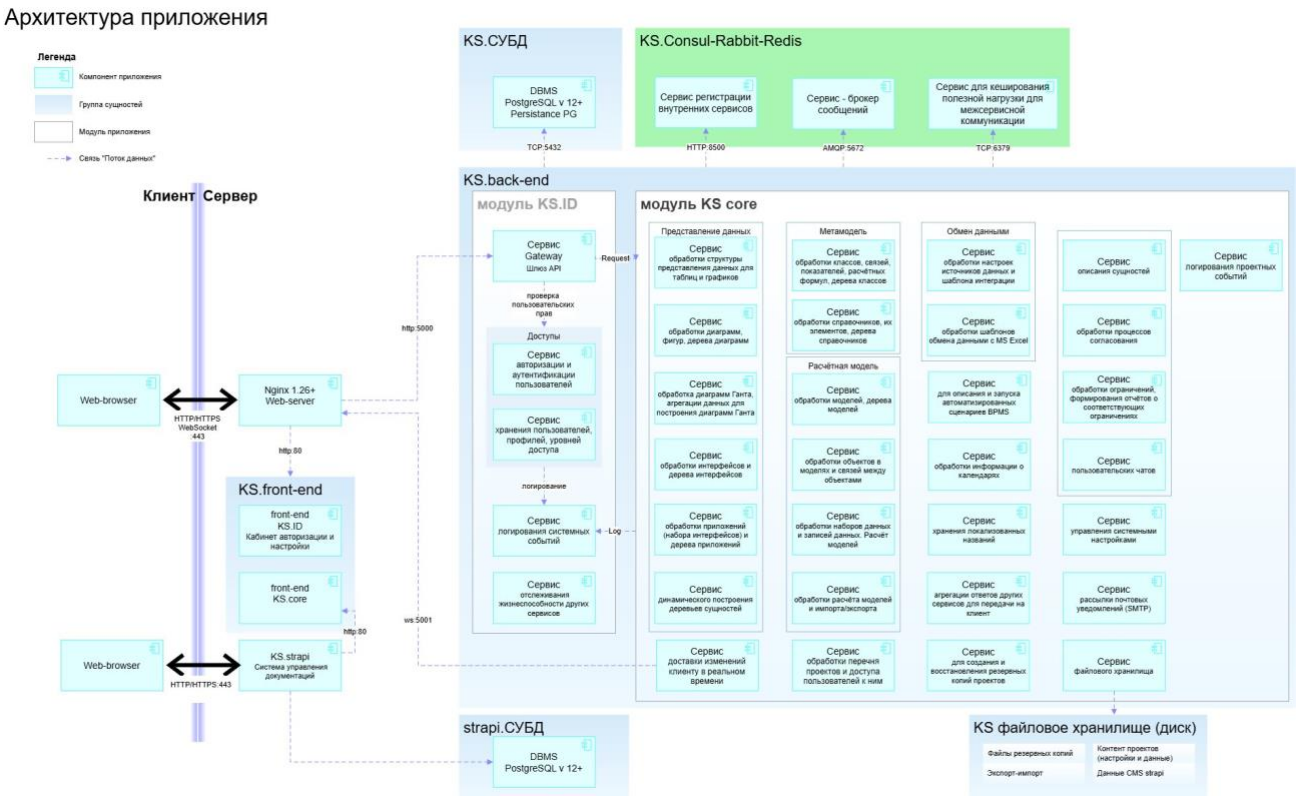

Рисунок 1 – Архитектура платформы Knowledge Space

Исходя из содержимого диаграммы можно сделать вывод, что Knowledge Space является многокомпонентным комплексом, включающим фронтенд, бэкенд, сервисы обработки сущностей, проектные сервисы, базы данных и файловое хранилище. Таким образом, изменение конфигурации затрагивает не один изолированный элемент, а совокупность взаимосвязанных компонентов, именно поэтому даже при использовании низкокодового подхода модификации должны проходить по регламентированной цепочке: от проектирования и настройки до размещения обновлённой версии в промышленной среде.

3. МОДЕЛЬ СРЕД DEV И PROD

Среда dev предназначена для проектирования, настройки и первичной проверки конфигурационных изменений. В этом контуре создаются новые сущности, корректируются бизнес-правила, редактируются формы, настраиваются показатели и другие элементы модели. Основная задача

разработческого окружения — предоставить безопасное пространство, в котором доработки не затрагивают текущую работу пользователей и могут свободно уточняться до нужного уровня зрелости.

Среда prod представляет собой эксплуатационный контур системы. Здесь размещается только утверждённая конфигурация, используемая в повседневной деятельности организации. Для рабочего окружения особенно важны стабильность, предсказуемое поведение и возможность быстрого восстановления после неудачного релиза [7; 9]. Поэтому вмешательство непосредственно в prod недопустимо: изменения должны поступать сюда из dev в виде согласованной и задокументированной версии.

Разделение двух контуров даёт ряд прикладных эффектов:

- снижает вероятность сбоев в работе пользователей при экспериментальных модификациях;
- обеспечивает фиксацию и сопоставление разных вариантов конфигурации;
- упрощает возврат к стабильному состоянию;
- делает процедуру развёртывания управляемой и повторяемой.

Для платформы Knowledge Space такая двухсредовая схема особенно значима, поскольку конфигурация формируется набором взаимосвязанных объектов, а не одним файлом. При отсутствии разделения сред даже локальное изменение может вызвать цепную реакцию в связанных сущностях и процессах.

4. ЖИЗНЕННЫЙ ЦИКЛ ИЗМЕНЕНИЯ КОНФИГУРАЦИИ

Для рассматриваемой платформы целесообразно использовать укрупнённый жизненный цикл изменения, включающий четыре этапа.

1. Инициирование изменения: на данной стадии формируется запрос на доработку или исправление конфигурации, определяется объект изменения и его ожидаемый результат.
2. Разработка в dev-среде: изменение реализуется в среде dev, где производится настройка сущностей и логика обработки данных.
3. Контроль версии и согласование: состояние конфигурации фиксируется в репозитории, проходит проверку и согласование ответственными участниками.
4. Развёртывание в prod**:** **у**тверждённая конфигурация переносится в промышленную среду, после чего выполняется контроль корректности её работы.

Данный цикл является укрупнённым и достаточным для описания управляемого переноса изменений в рамках статьи исследовательско-практического характера. Его преимуществом является простота и совместимость с классическими подходами к управлению релизами и конфигурациями [6; 9].

5. ИНТЕГРАЦИЯ GIT-СЕРВИСА, GITLAB И KNOWLEDGE SPACE

Для повышения управляемости изменений двухсредовая модель может быть усилена интеграцией платформы с Git-сервисом и GitLab. В этом случае Git используется как инструмент фиксации версий конфигурации, а GitLab — как средство централизованного хранения репозиториев, управления доступом и поддержки процессов согласования и автоматизации [6; 8].

Диаграмма, представленная на рисунке 2, иллюстрирует взаимодействие пользователя с веб-интерфейсом, посредством которого происходит управление проектами и запуск операций синхронизации. Внутри платформы выделены специализированные сервисы, а также базы данных проектов и пользователей.

Git Service играет ключевую роль: он отвечает за работу с токенами, операции push/pull, сериализацию и десериализацию данных в формате YAML и обмен с внешним GitLab через API. Такая организация подтверждает, что контур интеграции с Git может быть реализован как отдельный слой, не нарушающий базовую бизнес-логику платформы.

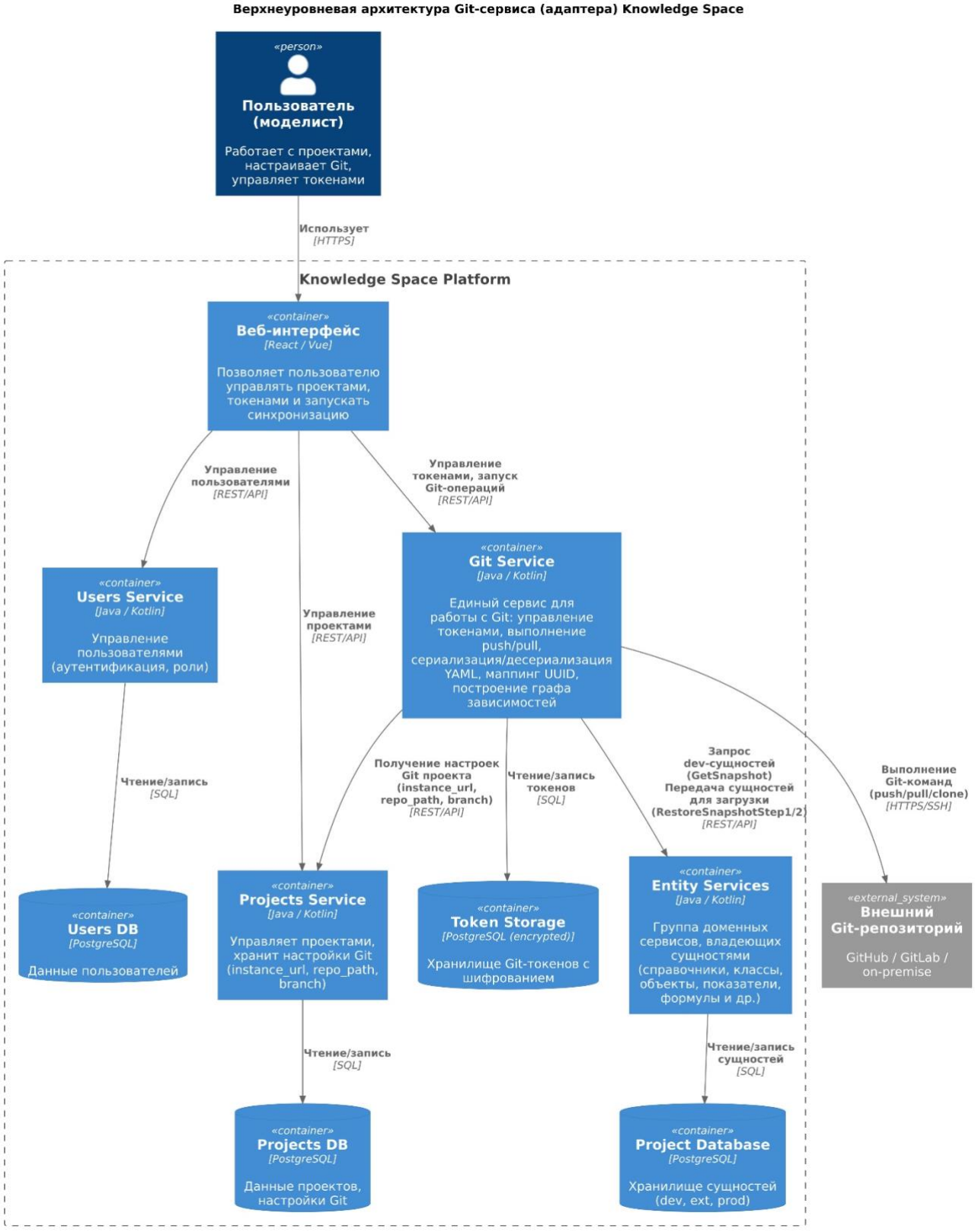

Рисунок 2 – Компонентная архитектура интеграции Git-сервиса, GitLab и Knowledge Space

С практической точки зрения последовательность выглядит следующим образом: изменение сначала разрабатывается в dev, затем его состояние передаётся в репозиторий, где фиксируется версия и выполняется контроль. После утверждения соответствующий вариант конфигурации используется для обновления prod. Тем самым Git и GitLab дополняют функционал Knowledge Space, повышая прозрачность, поддерживая коллективную работу и улучшая трассируемость конфигурационных изменений.

6. ПРАКТИЧЕСКОЕ ЗНАЧЕНИЕ ПОДХОДА

Предложенный подход имеет практическую ценность для организаций, использующих low-code платформу Knowledge Space для автоматизации внутренних процессов. Его применение позволяет сократить число ошибок, связанных с ручным изменением промышленной конфигурации, обеспечить единообразный порядок публикации изменений и повысить контролируемость сопровождения системы.

Кроме того, интеграция с Git-сервисом и GitLab создаёт основу для дальнейшего развития DevOps-практик: централизованного хранения версий, согласования изменений через merge request, построения сценариев автоматической проверки и последующего перехода к более зрелым CI/CD-процессам. Для корпоративных решений это особенно важно, поскольку стоимость ошибки в prod-среде существенно выше стоимости дополнительных процедур контроля на этапе разработки в окружении dev.

ЗАКЛЮЧЕНИЕ

В статье рассмотрена модель управления изменениями на платформе Knowledge Space, основанная на разделении сред dev и prod. Показано, что такая модель является необходимым условием безопасного сопровождения корпоративных low-code решений, поскольку позволяет изолировать разработку

от промышленной эксплуатации и обеспечить контролируемый перенос конфигурации.

Анализ архитектуры платформы и компонентной схемы интеграции с GitLab показывает, что Knowledge Space обладает необходимыми предпосылками для построения управляемого процесса версионирования и публикации изменений. Использование Git-сервиса и внешнего репозитория позволяет сделать жизненный цикл конфигурации более прозрачным, а работу с версиями — воспроизводимой и документируемой.

Практический результат исследования состоит в формулировке компактного и применимого подхода к управлению изменениями в low-code среде, который может использоваться как основа для дальнейшей формализации регламентов сопровождения и развития таких корпоративных информационных систем, как Knowledge Space.

СПИСОК ЛИТЕРАТУРЫ

1. Naqvi B., Kedziora D., Zhang L., Oyedeji S. Quality of Low-Code/No-Code Development Platforms Through the Lens of ISO 25010:2023 // Computer. 2025. Vol. 58. No. 3. P. 30–40.
2. Li J., Zhang Y., Wang X., Liu H. Democratizing Digital Transformation: A Multisector Study of Low-Code Adoption Patterns, Limitations, and Emerging Paradigms // Applied Sciences. 2025. Vol. 15. No. 12.
3. Rakovic L., Djordjevic Milutinovic L., Lula P., Vukovic V., Dziura M., Rojek T. Low-Code/No-Code Software Development: Definition, Drivers, and Inhibitors // Anali Ekonomskog fakulteta u Subotici. 2025. Vol. 61. No. 54. P. 1–18.
4. Аверина М. А., Ромашкин Е. А. Применение технологий low code и no-code при разработке программных продуктов // International Journal of Information and Communication Technologies. 2024. Т. 20. № 4. С. 58–71.

5. Кочеткова К. В., Осипова А. А., Зинина С. В. Low-code и no-code как инструменты трансформации бизнес-процессов // Управленческий учёт. 2024. № 6. С. 107–114.
6. Беляев Д. А., Смирнова Т. И. Управление версиями конфигурации Git в low-code-платформах // Системы управления и информационные технологии. 2024. № 2 (38). С. 37–48.
7. Техническая документация платформы Knowledge Space, 2024.
8. Чакон С., Страуб Б. Pro Git. Москва: ДМК-Пресс, 2019.
9. Forsgren N., Humble J., Kim G. Accelerate: The Science of Lean Software and DevOps. Portland: IT Revolution Press, 2018.

REFERENCES

1. Naqvi B., Kedziora D., Zhang L., Oyedeji S. Quality of Low-Code/No-Code Development Platforms Through the Lens of ISO 25010:2023. *Computer*, 2025, vol. 58, no. 3, pp. 30–40.
2. Li J., Zhang Y., Wang X., Liu H. Democratizing Digital Transformation: A Multisector Study of Low-Code Adoption Patterns, Limitations, and Emerging Paradigms. *Applied Sciences*, 2025, vol. 15, no. 12.
3. Rakovic L., Djordjevic Milutinovic L., Lula P., Vukovic V., Dziura M., Rojek T. Low-Code/No-Code Software Development: Definition, Drivers, and Inhibitors. *Anali Ekonomskog fakulteta u Subotici*, 2025, vol. 61, no. 54, pp. 1–18.
4. Averina M. A., Romashkin E. A. Application of low code and no-code technologies in software product development. *International Journal of Information and Communication Technologies*, 2024, vol. 20, no. 4, pp. 58–71.

5. Kochetkova K. V., Osipova A. A., Zinina S. V. Low-code and no-code as tools for business process transformation. *Management Accounting*, 2024, no. 6, pp. 107–214.
6. Belyaev D. A., Smirnova T. I. Git version control in low-code platforms. *Control Systems and Information Technologies*, 2024, no. 2 (38), pp. 37–48.
7. Knowledge Space technical documentation. 2024.
8. Chacon S., Straub B. *Pro Git*. Moscow: DMK Press, 2019.
9. Forsgren N., Humble J., Kim G. *Accelerate: The Science of Lean Software and DevOps*. Portland: IT Revolution Press, 2018.