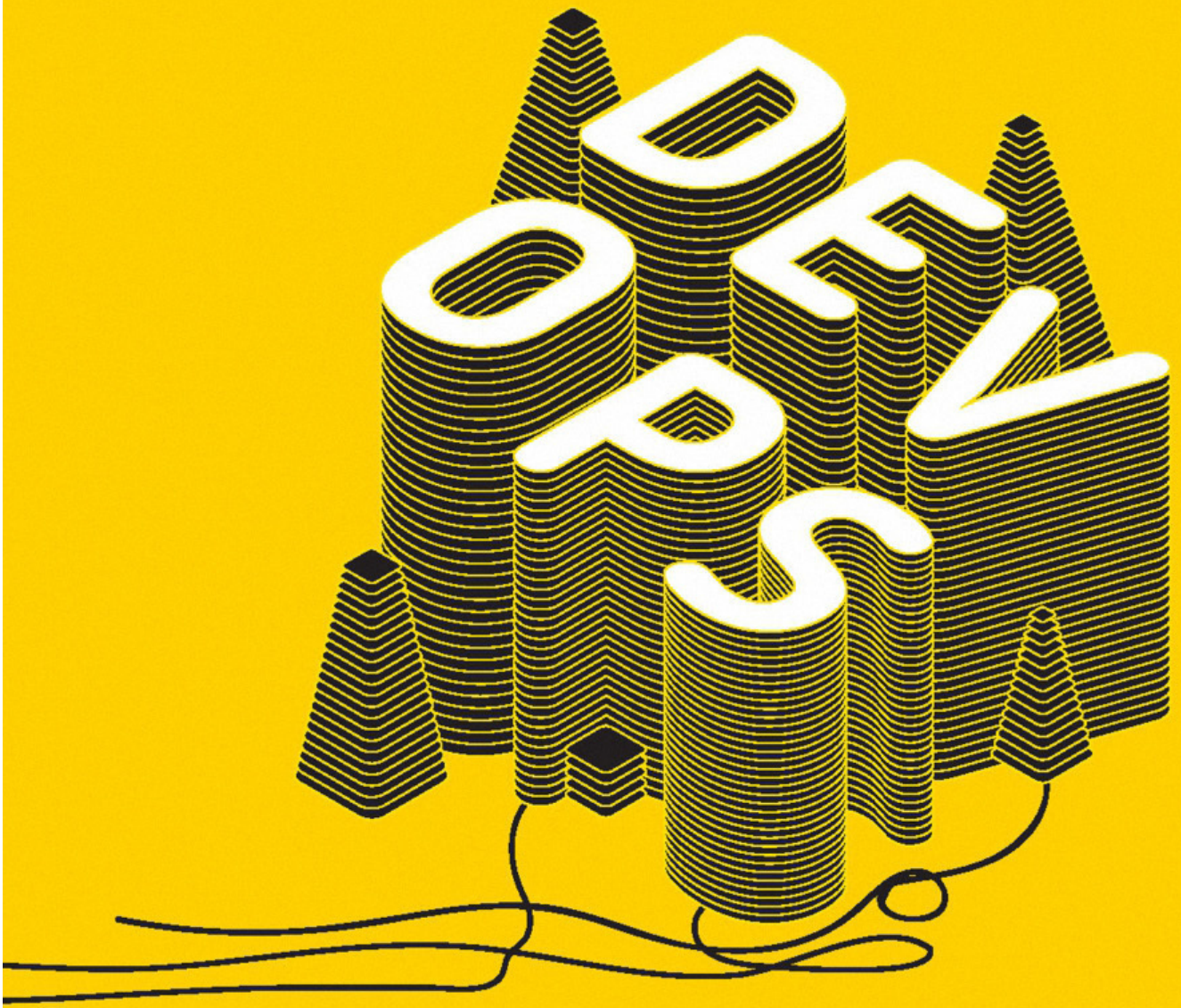
«Книга для тех, кто не боится инноваций и хочет вывести рабочие процессы компании на качественно новый уровень».

Елена Лукутина, директор по операционной деятельности и технологическому развитию «Неофлекс»

Руководство по DevOps

Джен Ким
Патрик Дебуа
Джон Уиллис
Джез Хамбл



Как добиться гибкости, надежности и безопасности мирового уровня в технологических компаниях

Джез Хамбл

Руководство по DevOps. Как добиться гибкости, надежности и безопасности мирового уровня в технологических компаниях

«Манн, Иванов и Фербер (МИФ)»

2016

УДК 658.81:004.4
ББК 65.290с51-32

Хамбл Д.

Руководство по DevOps. Как добиться гибкости, надежности и безопасности мирового уровня в технологических компаниях / Д. Хамбл — «Манн, Иванов и Фербер (МИФ)», 2016

ISBN 978-5-00100-750-0

Профессиональное движение DevOps зародилось в 2009 году. Его цель – настроить тесные рабочие отношения между разработчиками программного обеспечения и отделами IT-эксплуатации. Внедрение практик DevOps в повседневную жизнь организации позволяет значительно ускорить выполнение запланированных работ, увеличить частоту релизов, одновременно повышая безопасность, надежность и устойчивость производственной среды. Эта книга представляет собой наиболее полное и исчерпывающее руководство по DevOps, написанное ведущими мировыми специалистами.На русском языке публикуется впервые.

УДК 658.81:004.4
ББК 65.290с51-32

ISBN 978-5-00100-750-0

Содержание

Джин Ким, Патрик Дебуа, Джон Уиллис, Джез Хамбл Руководство по DevOps. Как добиться гибкости, надежности и безопасности мирового уровня в технологических компаниях

Информация от издательства

Научный редактор Николай Корытко

Издано с разрешения IT Revolution Press LCC c/o Fletcher & Company и Andrew Nurnberg Associates International Ltd c/o ZAO "Andrew Nurnberg Literary Agency"

Благодарим за помощь в подготовке издания Артема Каличкина, Дмитрия Зайцева, Михаила Чинкова, Виталия Рыбникова, Дениса Иванова, Валерия Пилия, Дмитрия Малыхина, Сергея Малютина, Александра Титова, Дениса Рыбака, Евгения Овчинцева, Алексея Климова, Игоря Авдеева

* * *

Предисловие к российскому изданию

Впервые о DevOps заговорили в связи с переходом в эру цифровой экономики, когда скорость выпуска на рынок продуктов стала одним из ключевых конкурентных преимуществ. Технологиям, обеспечивающим стремительное развитие бизнеса, пришлось бежать со всех ног, чтобы только оставаться на месте, а для достижения дополнительных результатов, как минимум, в два раза быстрее. Компаниям понадобились инструменты для быстрого и непрерывного улучшения качества существующих процессов разработки продуктов и их максимальной автоматизации, потому что хороший продукт стал равен хорошему ИТ.

Свой путь погружения в DevOps я начала несколько лет назад, когда возглавила отдел тестирования системы подготовки регулярной банковской отчетности Neoflex Reporting, которая отличалась большим количеством параллельных веток разработки и обилием ручных процессов. В ее разработку к этому моменту уже были вложены десятки тысяч человеко-часов.

Засучив рукава, наша команда взялась за точечную автоматизацию этапов жизненного цикла продукта. В целом мы достигли неплохих результатов, но добиться слаженной и синхронной работы от всех участников процесса оказалось по-настоящему трудной задачей. Периодически возникающие «тут подкрутить», «там вручную запустить», «а это не на моей стороне», «я был на обеде», «исторически сложилось» тормозили ожидаемое от автоматизации ускорение.

Осознать, что же делать дальше, помогла книга, которую вы сейчас держите в руках. Мы прочитали её всей командой и здорово переработали текущие процессы взаимодействия в парадигме слаженности, простоты и удобства. А процессы сборки, развертывания инфраструктуры, установки, тестирования и выдачи поставки объединили в непрерывный производственный конвейер, вдохновленные идеей «все, что связано с кодом – тоже код». Довольно быстро были получены ошеломляющие результаты: время выпуска обновлений с одного дня сократилось до десятка минут, а работа над продуктом Neoflex Reporting стала приносить профессиональное удовольствие.

«Руководство по DevOps» – книга об эффективном ИТ настоящего. Захватывающий и понятный путеводитель, способный обобщить, разложить по нужным полочкам существующий опыт и обогатить его ценными идеями.

В книге описаны основные шаги и принципы построения производственного взаимодействия, автоматизации процессов и развития культуры разработки ПО. Теория щедро сдобрена историями реальных людей и компаний, прошедших непростой, но интересный путь к DevOps.

Неоспоримая ценность «Руководства…» в том, что оно помогает вырваться из рутины бытия и взглянуть на текущие процессы совершенно другими глазами. Приходит осознание того, что на точечных «костылях» автоматизации далеко не уйти, появляется понимание того, как выглядит путь роста и развития, который подходит именно вашей компании, проекту, продукту.

Желаю вам приятного чтения и пусть эта книга станет для вас источником неиссякаемого вдохновения!

Лина Чуднова, руководитель практики DevOps компании «Неофлекс»

Введение

«Ага!»

Путь к созданию книги «Руководство по DevOps[1]» был долгим. Он начался в феврале 2011 г. с еженедельных переговоров по скайпу между соавторами. Мы решали, как создать руководство с рекомендациями – дополнение к книге The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win[2].

Прошло пять с лишним лет. Более двух тысяч часов работы. Книга «Руководство по DevOps» наконец завершена. В результате мы вознаграждены сполна, поскольку неожиданно обрели новое знание и поняли: сфера его применения гораздо шире, чем мы первоначально предполагали. Оно обеспечивает невиданные возможности. В конце концов мы сами воскликнули: «Ага!» – и нам кажется, что многие читатели разделят наше мнение.

Джин Ким

Мне повезло: с 1999 г. я изучал организации, использующие высокопроизводительные технологии. Вот один из моих первых выводов: решающее значение для успеха имеет перекрестное взаимодействие функциональных групп, занимающихся эксплуатацией, информационной безопасностью и разработкой. Я до сих пор помню, как впервые осознал масштабы нисходящей спирали, в которую заключена деятельность этих групп с их противоположными задачами.

Это было в 2006 г., и мне тогда представилась возможность поработать целую неделю с группой, решавшей отданные на аутсорсинг IT-задачи, поставленные крупной службой резервирования и продаж авиабилетов. Участники группы рассказали об увеличивающихся негативных последствиях ежегодных крупных обновлений программного обеспечения: каждый раз наступал настоящий хаос, шквал неудобств как для исполнителей, так и для заказчика. Из-за простоев у пользователей им приходилось выплачивать немалые компенсации согласно договорам по сервисному обслуживанию. Увольнялись наиболее способные и опытные работники, так как, опасаясь потерять прибыль, компания вынуждала их наращивать темп, выполнять массу незапланированной работы и «тушить пожары». У оставшегося персонала не хватало сил справляться со все возрастающим потоком требований заказчиков, желавших исправления ошибок. От расторжения сервисного контракта компанию спасали только героические усилия менеджеров среднего звена, и все были уверены: у контракта нет будущего, его не продлят на следующие три года.

Отчаяние и безнадежность подтолкнули меня к тому, чтобы начать нечто вроде наступательной операции. Разработка всегда рассматривалась как

[1] Акроним от англ. development и operations – методология разработки программного обеспечения, нацеленная на активное взаимодействие и интеграцию специалистов по разработке и специалистов по IT-обслуживанию. *Прим. перев.*

[2] *Ким Д., Бер К., Слаффорд Дж.* Проект «Феникс». Роман о том, как DevOps меняет бизнес к лучшему. М.: Эксмо, 2015. *Прим. перев.*

часть стратегии, а эксплуатация – тактики. Нередко они частично или даже полностью отдавались на аутсорсинг, чтобы лет через пять вернуться обратно, еще более усложнившимися.

Многие годы мы размышляли, как улучшить ситуацию. Вспоминаю, как на конференции Velocity Conference 2009 с интересом следил за обсуждением фантастических результатов, достигнутых благодаря использованию принципов бизнес-архитектуры, технических методов и норм корпоративной культуры в совокупности. Теперь эта методика известна нам как DevOps. Тогда я неподдельно взволновался: передо мной наметился путь выхода из создавшейся ситуации – его-то мы так долго искали. Стремясь распространить новое знание как можно шире, я и решил выступить соавтором The Phoenix Project. Вы запросто можете представить себе, какое огромное внутреннее удовлетворение я испытал, видя видя отзывы людей о том, как книга помогла им придти к озарению и воскликнуть: «Ага!»

Джез Хамбл

Мое личное «Ага!» впервые раздалось в 2000 г., в стартапе, который был моим первым местом работы после окончания обучения. Некоторое время нас, технических специалистов, было только двое. Поэтому мне приходилось заниматься всем: сетями, программированием, поддержкой пользователей, системным администрированием. Мы выпускали ПО, размещая его на FTP прямо с рабочих станций.

В 2004 г. я перешел в консалтинговую компанию ThoughtWorks, где впервые принял участие в работе над проектом в составе команды численностью около 70 человек. Я входил в группу из восьми инженеров, занимавшуюся развертыванием нашей программы в среде, приближенной к производственной. Поначалу задание вызывало у нас сильный стресс. Но спустя несколько месяцев мы перешли от режима работы вручную, занимавшего около двух недель, к автоматическому разворачиванию продолжительностью всего один час. Теперь можно было за доли секунды откатывать конфигурации назад и вперед, используя технику «Blue-Green разворачивания» в рабочее время[3].

Проект породил массу идей, изложенных как в этой книге, так и в другой – «Непрерывное развертывание ПО»[4]. Они убедили меня и многих моих коллег: с какими трудностями нам ни пришлось бы сталкиваться, мы способны действовать с максимальной отдачей и помогать людям.

Патрик Дюбуа

Для меня это была целая цепь событий. В 2007 г. я работал в проекте по миграции дата-центра совместно с несколькими Agile-командами. Я завидовал

[3] Техника «Blue-Green разворачивания» – стратегия установки ПО, базирующаяся на двух идентичных инсталляциях промышленной системы, одна из которых активна, и возможно мгновенное переключение между ними. Одна из них условно называется синей, ее копия же называется зеленой. *Прим. перев.*

[4] *Хамбл Д., Фарли Д.* Непрерывное развертывание ПО. Автоматизация процессов сборки, тестирования и внедрения новых версий программ… М.: Вильямс, 2011. *Прим. перев.*

их высокой продуктивности, умению выполнять большой объем работы за ограниченное время.

Получив следующее задание, я приступил к эксперименту по внедрению методики канбан в работу группы эксплуатации и увидел: группа стала быстро меняться. Позже, на конференции Agile Toronto 2008, я представил свой доклад в IEEE[5] на эту тему, но, к сожалению, он не получил широкого отклика в Agile-сообществе. Мы начали создавать административную группу для системы Agile, но тут я переоценил значение человеческого фактора.

Увидев на Velocity Conference 2009 презентацию Джона Олспоу и Пола Хаммонда «10 развертываний в день», я убедился: у меня есть единомышленники. Поэтому я решил организовать первую конференцию DevOpsDays и так случайно создал новый термин – DevOps.

Энергетика мероприятия была уникальной. Участники благодарили меня, утверждая, что конференция изменила их жизнь к лучшему. Так я осознал степень воздействия концепции DevOps и с тех пор неустанно продвигаю ее.

Джон Уиллис

В 2008 г. я продал свою консалтинговую компанию, специализировавшуюся на внедрении крупномасштабных устаревших решений в области управления конфигурациями и мониторинга (Tivoli). Тогда же я впервые встретил Люка Каниса (основателя компании PuppetLabs). Люк выступал с презентацией о Puppet на проводившейся издательством O'Reilly конференции по конфигурационному управлению (CM) на основе открытого исходного кода.

Поначалу я скучал на заднем ряду лекционного зала, размышляя, что нового этот двадцатилетний парень может рассказать мне об управлении конфигурациями. Ведь я занимался этим всю жизнь, помогая крупнейшим корпорациям мира разрабатывать решения в области CM и других сферах управления эксплуатацией. Однако через пять минут после начала доклада я уже сидел на первом ряду. Я тут же понял: все, что я делал за последние 20 лет, я делал неправильно. Люк описывал то, что я сейчас называю вторым поколением CM.

После доклада мне удалось поболтать с ним за чашечкой кофе. Я был совершенно восхищен идеей, сейчас называемой «инфраструктура как код». Люк увлекся и стал подробно объяснять, что имеет в виду. Он верит, что эксплуатация становится похожей на разработку программ. Специалисты отдела эксплуатации хотят, чтобы конфигурации проверялись системой контроля качества и чтобы в рабочий процесс были адаптированы методики обеспечения CI/CD[6]. Поскольку я к тому времени уже немало проработал в области эксплуатации IT, я ответил ему примерно так: «Это то же, что пытаться заставлять управленцев петь как Led Zeppelin».

Я жестоко ошибался.

---

[5] Институт инженеров электротехники и электроники (от *англ.* Institute of Electrical and Electronics Engineers) – международная некоммерческая ассоциация специалистов в области техники, мировой лидер в области разработки стандартов по радиоэлектронике, электротехнике и аппаратному обеспечению вычислительных систем и сетей. *Прим. перев.*

[6] Continuous Integration / Continuous Deployment – непрерывная интеграция и непрерывное развертывание. *Прим. ред.*

> Примерно через год на другой конференции Velocity, проводившейся в 2009 г. O'Reilly, я увидел презентацию Эндрю Шефера по инфраструктуре Agile. В ней он показал ставший каноническим рисунок – метафорическую стену между разработчиками и инженерами эксплуатации, через которую они перебрасывают друг другу рабочие задания. Он назвал это «стеной неразберихи». Идеи, высказанные в презентации, в систематизированном виде выражали то, что Люк пытался рассказать мне годом ранее. Для меня это стало откровением. В том же году меня, единственного из американцев, пригласили на первую конференцию DevOpsDays в Генте. Ко времени окончания конференции идея, ныне получившая название DevOps, полностью овладела моим разумом.

Из всего вышесказанного ясно, что соавторы книги пришли к единому выводу, хотя шли к нему разными путями. Реальность доказала, что описанная проблема существует практически везде и решения с помощью DevOps применимы повсюду.

Цель написания этой книги – показать, как воспроизвести DevOps-трансформации, частью которых мы были или которые мы наблюдали со стороны. Еще мы хотим развеять множество мифов о том, почему DevOps не будет работать в тех или иных ситуациях. Нам довелось слышать примерно следующее.

Миф 1: *DevOps пригоден только для стартапов*. Методы DevOps впервые были применены единорогами интернет-индустрии: Google, Amazon, Netflix и Etsy. Каждая из компаний в определенные моменты своей истории рисковала выпасть из бизнеса из-за проблем, обычно возникающих в традиционных организациях (их еще называют рабочими лошадками экономики). Это опасные релизы, приводящие компанию к катастрофическому провалу, неумение быстро проводить изменения продукта или сервиса, чтобы превзойти конкурентов в новой области, проблемы с соблюдением нормативных требований, неспособность масштабироваться, высокая степень недоверия между разработкой и эксплуатацией и так далее.

Однако каждая из названных организаций смогла преобразовать свою архитектуру, технические методы, производственную культуру и достичь выдающихся результатов благодаря DevOps. Как язвительно заметил известный американский специалист по информационной безопасности Бранден Вильямс, «пусть больше не будет разговоров о пегасах или рабочих лошадках в DevOps, пусть останутся только чистокровные скакуны, а остальные отправятся на мыловарню в качестве сырья».

Миф 2: *DevOps заменяет собой Agile.* Принципы и методы DevOps совмещаются с Agile, причем многие отмечают, что DevOps – логическое продолжение Agile. Agile часто оказывается эффективным катализатором DevOps, поскольку предпочитает организовывать деятельность небольших команд, непрерывно поставляющих пользователям код высокого качества.

Многие практики DevOps возникают, если мы продолжаем управлять нашей работой за пределами цели «код, потенциально пригодный для релиза» в конце каждой итерации, расширяя ее до того, чтобы наш код всегда находился в развертываемом состоянии, а разработчики ежедневно синхронизировали свои изменения с основной веткой кода и могли продемонстрировать новые изменения в окружениях, близким к реальным.

Миф 3: *DevOps несовместим с ITIL.* Многие рассматривают DevOps как ответ на ITIL или ITSM (управление IT-инфраструктурой компании). Его описание было впервые опубликовано в 1989 г. ITIL сильно повлияла на несколько поколений практиков в области управления инфраструктурой, включая одного из авторов этой книги. Это постоянно развивающаяся библиотека методов, позволяющих кодифицировать процессы и практики, лежащие в основе

признанных во всем мире способов управления IT, связывающих воедино стратегию услуг, разработку и поддержку.

Методы DevOps можно сделать совместимыми с процессами ITIL. Однако для обеспечения более короткого цикла разработки и повышения частоты развертываний, предложенных DevOps, многие области процессов ITIL должны быть полностью автоматизированы. Следует решить проблемы, связанные с процессами управления конфигурациями и релизами (например, как обеспечивать постоянную готовность базы управления и библиотеки программ). DevOps требует, чтобы возникающие ошибки быстро обнаруживались и устранялись, поэтому дисциплина применения ITIL в проектировании архитектуры, обработке сбоев, решении проблем остается актуальной, как и всегда.

Миф 4: *DevOps несовместим с требованиями информационной безопасности.* Отсутствие традиционных методов контроля (например, разделение ответственности, изменение процессов проверки кода, проверка безопасности вручную по окончании проекта) может вызвать тревогу у специалистов по безопасности.

Однако это не означает, что в организациях, использующих DevOps, отсутствует эффективный контроль. Вместо работ по обеспечению безопасности и проверки соответствия требования ИБ, проводящихся на завершающем этапе проекта, необходимые проверки интегрированы в каждую стадию ежедневного цикла на протяжении всего цикла разработки. В результате обеспечиваются более высокое качество и безопасность.

Миф 5: *DevOps означает отсутствие необходимости управления IT-эксплуатацией, то есть NoOps (дословно – «нет эксплуатации»).* Многие неправильно трактуют DevOps как полное исключение необходимости IT-эксплуатации. Однако такое утверждение редко бывает справедливо. Хотя характер оперирования может измениться, само управление остается важным как никогда. Просто оно на гораздо более ранних этапах жизненного цикла ПО взаимодействует с разработкой, продолжающей действовать параллельно с IT-эксплуатацией еще долго после того, как разработанный код развернут в производственной среде.

Вместо того чтобы отдел эксплуатации разгребал поступающие заявки вручную, DevOps дает разработчикам возможность делать большинство операций через API и самообслуживающиеся платформы, такие как: создание среды, тестирование и развертывание кода, получение и отображение метрик о ПО и т. д. Когда реализован такой подход, IT-эксплуатация становится похожей на процесс разработки (то же справедливо для управления качеством и обеспечения информационной безопасности), сцепленный с разработкой продукта, где под продуктом понимается платформа, используемая, чтобы надежно, быстро и безопасно тестировать IT-сервисы, развертывать их и запускать в производственной среде.

Миф 6: *DevOps – это просто реализация подхода «инфраструктура как код».* Хотя многие из практик и подходов DevOps, приведенных в этой книге, требуют автоматизации, для реализации DevOps также необходимо изменение архитектуры системы и культуры производства, дающее возможность достичь общих целей в ходе работы по повышению создаваемой ценности IT. Это выходит далеко за рамки простой автоматизации. Как написал Кристофер Литл, один из самых первых летописцев DevOps, «это не автоматизация, так же как астрономия – это не телескопы».

Миф 7: *DevOps применим только к программам с открытым исходным кодом.* Хотя многие случаи успешного внедрения DevOps действительно имели место в организациях, использовавших ПО, входившее в группу LAMP (Linux, Apache, MySQL, PHP), имелись истории успеха, и не зависевшие от использованных технологий. Например, в приложениях, напи-

санных на Microsoft.NET, коболе, языке ассемблера мейнфреймов, а также в системах SAP и даже в коде встроенных систем (например, микропрограммное обеспечение принтеров HP LaserJet).

«Ага!» еще несколько раз

Каждый из авторов воодушевился удивительными инновациями, реализованными сообществом DevOps, и достигнутыми результатами: были созданы надежные системы, позволившие небольшим командам быстро и независимо разрабатывать и проверять код, который может быть без риска развернут у заказчика. Учитывая нашу убежденность, что DevOps – воплощение методов создания динамичных, обучающихся организаций, постоянно укрепляющих культуру высокого доверия, представляется неизбежным, что эти организации будут продолжать инновации и выйдут победителями на рынке.

Мы искренне надеемся, что книга «Руководство по DevOps» станет ценным источником. Как руководство по проведению DevOps-трансформации. Как набор практических примеров для накопления опыта. Как летопись истории DevOps. Как средство для организации коалиции и достижения общих целей владельцев продукта, архитекторов, разработчиков, инженеров контроля качества, эксплуатации и информационной безопасности. Она подскажет, как получить максимальную поддержку со стороны руководства при внедрении инициатив DevOps, как сформировать нравственный императив для изменения способов управления технологическими организациями при обеспечении высокой эффективности. Она поможет создать более оживленную и дружелюбную рабочую среду, чтобы любой участник смог учиться в течение всей жизни – это не только поможет каждому исполнителю достичь целей, но и приведет организацию к победе.

Предисловие

В прошлом многие сферы инженерной деятельности пережили серьезные изменения, за счет чего она сама стала понятнее. И хотя существуют университетские курсы и компании, осуществляющие техническую поддержку в каждой конкретной области (гражданском строительстве, механике, электроснабжении, атомной технике и т. д.), современное общество нуждается во всех перечисленных направлениях, чтобы по достоинству оценить их полезность и развивать междисциплинарные направления.

Вот, например, конструирование современного автомобиля. Где кончается область компетентности инженера-механика и начинается зона ответственности инженера-электрика? Где (и как, и когда) специалист по аэродинамике (безусловно, имеющий четкое представление о форме, размере и местах размещения окон) должен взаимодействовать с экспертом в области эргономики пассажиров? Что можно сказать о химическом влиянии топливной смеси и масел на материалы, из которых изготовлены двигатель и трансмиссия, в течение всего времени эксплуатации машины? Во время конструирования автомобиля можно задать много других вопросов, но конечный результат одинаков: для успеха современных технических проектов абсолютно необходимы две вещи – умение рассмотреть проблему с разных точек зрения и опыт совместной деятельности.

Чтобы направление деятельности или область знаний развивались и крепли, необходимо достичь того уровня, когда появляется возможность серьезно задуматься, как зародилось это направление или область, отыскать перспективы развития и объединить все это, желая понять облик общества будущего.

В этой книге представлен способ такого синтеза. Ее следует рассматривать как стартовый набор перспектив в области разработки и эксплуатации ПО (я по-прежнему считаю эту сферу деятельности растущей и быстро развивающейся).

Независимо от того, в какой отрасли вы работаете, какой продут выпускаете, какие услуги оказывает ваша организация, данный образ мышления имеет первостепенное значение: он просто необходим для выживания любой компании-лидера в любой области бизнеса и технологий.

Джон Олспоу, директор по технологиям компании Etsy
Бруклин, Нью-Йорк, август 2016 г.

Вступление. Как будет выглядеть мир, если разработка и эксплуатация пойдут по принципу DevOps

Представьте себе мир, в котором владельцы продукта, разработчики, тестировщики, сотрудники IT-эксплуатации и специалисты по информационной безопасности действуют сообща, не только помогая друг другу, но и обеспечивая будущий успех организации. Трудясь ради общей цели, они обеспечивают быстрое внедрение плановых результатов в производство (выполняя в день десятки, сотни или даже тысячи развертываний кода) и достигают при этом высокого уровня стабильности, устойчивости, доступности и безопасности.

В таком мире кросс-функциональные группы, несомненно, выполняют проверку своих предположений, какие именно функции особенно порадуют пользователей и послужат достижению целей организации. Они не просто заботятся о реализации функций, нужных пользователям, но также активно обеспечивают бесперебойную работу и проверку цепочки создания ценностей, не вызывая при этом хаоса в управлении IT-эксплуатацией и сбоев у любых внутренних и внешних клиентов.

В то же самое время тестировщики, сотрудники эксплуатации и специалисты по информационной безопасности постоянно стараются уменьшить взаимные трения внутри команды разработчиков, создавая системы, дающие возможность действовать продуктивнее и результативнее. Когда компетентность специалистов по качеству, сотрудников IT-эксплуатации и специалистов по информационной безопасности становится доступной командам, занимающимся поставками, автоматизированными инструментами и платформами самообслуживания, они могут использовать все это в повседневной работе и перестать зависеть от других команд.

Такой подход дает организациям возможность создать надежную систему: небольшие команды быстро и автономно разрабатывают, тестируют и развертывают код и вдобавок делают это надежно и безопасно. Это позволяет организациям максимизировать продуктивность разработчиков, организовать обучение внутри организации, обеспечить высокую удовлетворенность исполнителей и стать победителем в конкурентной рыночной борьбе.

Таковы результаты использования DevOps. Но большинство из нас живет совсем в другом мире. Зачастую системы, в которых мы работаем, несовершенны, демонстрируют очень низкие результаты и не позволяют раскрыть наш истинный потенциал. В нашем мире отделы разработки и IT-эксплуатации враждуют; тестирование и обеспечение информационной безопасности проводятся только ближе к окончанию проекта, то есть слишком поздно для устранения проблем. И почти любая серьезная деятельность требует от сотрудника значительных усилий, выполнения вручную ряда последовательных задач, из-за чего занятые поиском и устранением проблем инженеры и конечные пользователи просто ждут, пока кто-то сделает свою часть. Это не просто замедляет получение результата, но и вносит хаос, особенно в процесс развертывания, что приводит к негативным последствиям как для клиентов, так и для бизнеса.

В результате мы оказываемся далеко от желаемых целей. Все в организации недовольны уровнем производительности IT-подразделений. Итог – бюджет урезан, все разочарованы, но понимают, что бессильны изменить ход процесса разработки и добиться заметных результатов[7]. Каким же должно быть решение? Необходимо изменить методы работы, и DevOps – это наилучший способ продвижения вперед.

Чтобы лучше понять потенциал DevOps, давайте рассмотрим промышленную революцию 1980-х гг. Внедрив принципы и методы бережливого производства (Lean), промышленные

[7] Это всего лишь небольшой пример проблем, встречающихся в типичной IT-организации. *Прим. авт.*

компании значительно улучшили производительность предприятий, сократили время выполнения заказов, повысили удовлетворенность своих потребителей, что позволило им стать победителями на рынке.

Раньше среднее время выполнения заказа заводом-изготовителем составляло шесть недель, причем без нарушения сроков выполнялось менее 70 % заказов. К 2005 г. благодаря широкому внедрению методов бережливого производства временной показатель сократился менее чем до трех недель, и 95 % заказов выполнялись точно в срок. Организации, не внедрившие принципы бережливого производства, потеряли долю на рынке, а многие вообще ушли из бизнеса.

Точно так же повысились требования к продуктам: то, что было хорошо в предшествующие десятилетия, перестало удовлетворять заказчиков. Следующие 40 лет стоимость и время разработки и внедрения стратегических возможностей для бизнеса снижались все активнее. В 1970–1980-х гг. разработка и внедрение большинства новых технологий требовали от одного до пяти лет и часто обходились в десятки миллионов долларов.

К 2000-м гг. благодаря развитию технологий и внедрению принципов и методов Agile время, необходимое для разработки новой функциональности, уменьшилось до месяцев или даже недель. Но и сам процесс внедрения также занимал недели или месяцы, причем часто с неприемлемыми результатами.

Но к 2010 г. в связи с внедрением DevOps и непрекращающейся коммодитизацией[8] компьютерного оборудования, программ, а теперь и облачных технологий новые функции (и даже целые компании-стартапы) могут создаваться за недели и быстро – за часы или даже за минуты – внедряться в производство. Для таких организаций развертывание стало рутинной операцией, практически не содержащей рисков. Появилась возможность проводить эксперименты для проверки бизнес-идей, выясняя, какие из них наиболее ценны для клиентов и самой организации и какие можно быстро превратить в «фичи», а уже их, в свою очередь, быстро и без рисков развернуть на производстве.

Таблица 1. Тенденция к более быстрой, дешевой и имеющей мало рисков поставке программного обеспечения

[8] Коммодитизация – превращение изделий в обезличенный товар. *Прим. перев.*

	1970–1980 гг.	1900-е гг.	С 2000-х гг. до наших дней
Эпоха	Мейнфреймы	Клиент-сервер	Коммодитизация и облака
Типичные технологии эпохи	COBOL, DB2 в MVS и пр.	C++, Oracle, Solaris и т.д.	Java, MySQL, Red Hat, Ruby on Rails, PHP и т.д.
Продолжительность цикла разработки и внедрения	1–5 лет	3–12 месяцев	2–12 недель
Стоимость, млн долл. США	1–100	0,1–10	0,01–1
Зона риска	Вся компания	Линейка продуктов или отдел компании	Функциональность продукта
Цена ошибки	Банкротство, продажа компании, массовые сокращения	Потеря прибыли, увольнение руководства	Пренебрежимо мала

Источник: Презентация «Быстрота и объем (Скорость выигрывает)» Адриана Кокрофта на конференции FlowCon в Сан-Франциско, ноябрь 2013 г.

Организации, внедрившие принципы и методы DevOps, сейчас нередко за день выполняют сотни, а то и тысячи развертываний. В эпоху, когда для получения конкурентных преимуществ требуется быстрый выход на рынок и непрекращающееся экспериментирование, компаниям, неспособным показать такие же результаты, суждено уступить свою долю рынка более гибким и легким на подъем конкурентам и даже полностью уйти из бизнеса, подобно промышленным предприятиям, своевременно не внедрившим принципы бережливого производства.

Сегодня независимо от того, к какой отрасли относится компания, приобретение клиентов и предоставление им создаваемой ценности зависит от технологичности канала поставки этой ценности. Джеффри Иммелт, исполнительный директор компании General Electric, выразил эту мысль сжато и точно: «Каждая отрасль промышленности и каждая компания, не желающие делать программное обеспечение центром бизнес-стратегии, не имеют будущего». Или, как сказал Джеффри Сновер, технический специалист корпорации Microsoft, «в предыдущие экономические эпохи коммерческие предприятия создавали ценности, перемещая атомы. Теперь они делают это, перемещая биты».

Трудно переоценить масштабы этой проблемы. Она есть в каждой организации независимо от отрасли, размера компании и ее профиля – коммерческого или нет. Сейчас чаще чем

когда-либо то, как идет управление и выполняется технологическая работа, определяет главное: победит ли организация на рынке и, более того, выживет ли она. Во многих случаях нам придется освоить принципы и методы, на первый взгляд разительно отличающиеся от тех, которыми мы успешно руководствовались в предыдущие десятилетия (см. приложение 1).

Итак, теперь, когда мы обосновали неотложность проблемы, давайте уделим некоторое время более подробному изучению ее симптомов и тому, почему она возникает и почему с течением времени только обостряется, если не принять серьезные меры.

Проблема: кое-что в вашей организации должно быть улучшено (иначе вы не стали бы читать эту книгу)

Большинство компаний не в состоянии развернуть производственные изменения в реальной среде в течение нескольких минут или часов, им для этого потребуются недели или даже месяцы. Они также не могут доставлять в рабочую среду сотни или тысячи изменений в день, вместо этого они изо всех сил пытаются произвести развертывание ежемесячно или хотя бы ежеквартально. Также не предусматривается развертывание производственных процессов, вместо этого – перебои в работе и хронически героические усилия, направленные на ликвидацию проблем.

В эпоху, когда для получения конкурентного преимущества надо быстро выпускать продукты на рынок, обеспечивать высокий уровень поддержки и неустанно экспериментировать, такие организации оказываются в невыгодном для конкуренции положении. В значительной степени это обусловлено неспособностью разрешить коренной, хронический конфликт в их технологической организации.

Коренной, хронический конфликт

Почти в любой IT-компании существует постоянный конфликт между разработкой и IT-эксплуатацией, что создает нисходящую спираль и приводит к постоянному увеличению времени, необходимого для выпуска на рынок новых продуктов или новых функциональностей, снижению качества и, что самое плохое, к постоянному увеличению технического долга[9].

Термин «технический долг» был впервые предложен Уордом Каннингемом. Подобно финансовому, технический долг – решения, необходимые для ликвидации проблем, с течением времени становящихся все более трудно разрешимыми при постоянном уменьшении будущих возможностей для маневра. Даже действуя благоразумно, мы все равно вынуждены выплачивать проценты.

Один из факторов, вызывающих такое состояние, – часто проявляющаяся конкуренция между разработчиками и IT-эксплуатацией. Организации, специализирующиеся в области информационных технологий, должны отвечать за многое. Среди прочих есть две цели, и они должны достигаться одновременно:

- реагировать на быстро меняющийся конкурентный ландшафт;
- обеспечить стабильный, надежный и безопасный сервис для клиента.

Нередко разработчики берут на себя ответственность за реагирование на изменения на рынках, развертывание новой функциональности и изменений в реальной среде в кратчай-

[9] Термин из области разработки ПО: как только появляются изменения в программном коде, зачастую возникает необходимость сделать связанные с ними изменения в других частях кода или документации. Эти необходимые, но незавершенные изменения считаются долгом. Он должен быть погашен в определенный момент в будущем. *Прим. перев.*

шие сроки. Отдел IT-эксплуатации готов отвечать за предоставление заказчикам стабильных, надежных и безопасных IT-услуг, делая при этом затруднительным или даже невозможным внесение каких-либо изменений, ставящих производство под угрозу. При такой ситуации разработчики и отдел IT-эксплуатации преследуют абсолютно противоположные цели и имеют разные стимулы.

Доктор Элияху Голдратт, один из создателей методологии управления производством («теории ограничений»), называл конфигурации такого типа «корневым, хроническим конфликтом». Он заключается в том, что корпоративное нормирование и стимулы в разных подразделениях препятствовали достижению глобальных целей организации [10].

Этот конфликт настолько сильно препятствует результативности в бизнесе, как внутри IT-организаций, так и вне их, что возникает нисходящая спираль. Хронические конфликты зачастую ставят технических специалистов в условия, приводящие к созданию негодного ПО, низкому качеству поддержки, плохим результатам у заказчиков. А еще к тому, что практически ежедневно приходится искать обходные пути для решения проблем и незамедлительно прилагать героические усилия по внесению исправлений в отделах управления производством, разработки, тестирования, IT-эксплуатации или информационной безопасности (см. приложение 2).

Нисходящая спираль: драма в трех актах

У этой драмы три акта, вероятно, знакомые всем, кто имеет отношение к сфере IT.

Первый акт начинается в отделе IT-эксплуатации, где главная цель – сохранение работоспособности приложений и инфраструктуры, чтобы организация могла передавать клиентам продукцию. В повседневной деятельности многие проблемы вызваны тем, что приложения и инфраструктура оказываются сложными, плохо документированными и невероятно хрупкими. Это технический долг и ежедневный поиск обходных решений. С ними приходится постоянно сталкиваться, выслушивая клятвы, что кавардак будет обязательно ликвидирован, как только появится немного свободного времени. Но таковое, естественно, никогда не появляется.

Вызывает тревогу то обстоятельство, что наиболее хрупкие творения обеспечивают поддержку наиболее важных систем, создающих доход, или особенно серьезных проектов. Иными словами, именно системы, чаще всего выходящие из строя, наиболее важны для нас, и, кроме того, именно на них сильнее всего сказываются внесенные нами срочные изменения. И когда эти корректировки приводят к сбою, они ставят под удар наиболее важные обязательства организации, такие как доступность для клиентов, цели по получению дохода, безопасность данных пользователей, точность финансовых отчетов и так далее.

Второй акт начинается, когда кто-то должен компенсировать невыполненные обязательства – это может быть менеджер продукта, обещающий реализовать более впечатляющие возможности, чтобы восхитить заказчиков, или директор, предлагающий более агрессивные цели по доходности. Затем, забыв о реальных возможностях технологии или о том, из-за каких обстоятельств не были выполнены предыдущие обязательства, руководство заставляет технические отделы любой ценой реализовать эти новые обещания.

В результате перед разработчиками ставится задача срочно создать проект, что неизбежно требует решения новых технических проблем и «удаления всего лишнего», чтобы успеть реализовать задачу в срок. Так увеличиваются обязательства по техническому долгу, а парал-

[10] В мире промышленного производства существует похожий корневой, хронический конфликт. Его суть в необходимости обеспечивать своевременные поставки клиентам и управление затратами. Как этот конфликт был преодолен, рассказывается в приложении 2. *Прим. авт.*

лельно звучат привычные обещания исправить все проблемы, как только появится немного свободного времени.

Такое решение подготавливает сцену для третьего и последнего акта пьесы, когда все необходимые действия становятся более и более трудными: каждый исполнитель все глубже увязает в выполнении задания, коммуникации между сотрудниками и отделами становятся медленными, а очередь из заданий на выполнение понемногу растет. Работа затягивается узлом, небольшие действия вызывают большие сбои, все начинают проявлять опасение и нетерпимость к изменениям. Нужно больше обсуждений, координации и одобрения различными инстанциями. Группы сотрудников чаще вынуждены ждать, когда будет выполнена задача, задерживающая их собственные действия, а качество при этом только ухудшается. Дело движется все медленнее, и, чтобы увеличить скорость, приходится прилагать больше усилий (см. приложение 3).

В настоящем разглядеть нисходящую спираль сложно, но ретроспективно она видна отчетливо. Мы начинаем замечать, что развертывание готового кода занимает все больше времени: вместо минут – часы, дни и даже недели. Но хуже всего то, что результаты развертывания оставляют желать лучшего, а это ведет к возрастанию времени простоев у клиентов, что, в свою очередь, требует от отдела IT-эксплуатации героических усилий по «тушению пожаров», отнимающих у него возможность выплатить технический долг.

В результате циклы поставки продукта затягиваются все сильнее, новых начинаний все меньше, а проекты, которые все-таки появляются, оказываются менее амбициозными. Кроме того, обратная связь, доносящая результаты от каждого – особенно поступающая от клиентов, – ослабевает и становится более медленной. Несмотря на все усилия, ситуация только ухудшается: мы уже не в состоянии быстро реагировать на изменяющуюся ситуацию на рынке, предоставляя стабильный и надежный сервис клиентам. В результате мы теряем свою долю рынка.

Снова и снова повторяется одно и то же: если терпит неудачу подразделение IT, то провал ждет всю организацию. Как пишет в своей книге The High-Velocity Edge Стивен Спир, неважно, наступают ли печальные последствия «медленно, как постепенно развивающаяся болезнь», или происходят быстро, «как пожар дома… разрушение в обоих случаях полное».

Почему нисходящая спираль встречается везде

Более десяти лет авторы этой книги наблюдали нисходящую спираль в огромном количестве организаций всех типов и размеров. И в конце концов поняли, из-за чего она появляется и почему для ее устранения необходимо использование принципов DevOps. Во-первых, как уже говорилось, каждая IT-компания имеет две противоположные цели, а во-вторых, любая такая организация – технологическая, понимает она это или нет.

По словам Кристофера Литла, одного из первых летописцев DevOps, «каждая компания – технологическая, независимо от того, к какой области бизнеса она себя причисляет. Банк – это просто IT-организация с банковской лицензией»[11].

Чтобы убедиться в верности этих слов, примите во внимание, что большинство крупных проектов связаны со сферой IT. Как говорится, почти невозможно принять какое-либо бизнес-решение, не вызывающее хотя бы одного изменения в работе IT-подразделений.

В деловом и финансовом мире проекты важны, поскольку служат основными двигателями изменений внутри организаций. Проекты должны получить одобрение руководства, для них утверждается бюджет, за расходы нужно отчитываться, поэтому проекты – способ осуществить любые цели и ожидания организации, то есть и роста, и сокращения[12].

---

[11] В 2013 году в европейском банке HSBC трудилось больше разработчиков ПО, чем в компании Google. *Прим. авт.*

[12] Пока не будем устраивать дискуссий, как должна финансироваться разработка ПО, как «проект» или как «продукт».

Проекты обычно финансируются за счет вложения капитала (например, заводы: закупка оборудования, главные части проектов и расходы капитализируются и окупятся спустя годы). 50 % в наше время тратится на технические нужды. Это справедливо даже для вертикальных линеек так называемой низкотехнологичной промышленности, то есть тех отраслей, где исторически сложилось так, что вклады в разработку технологий невысоки (энергетика, металлургия, ресурсодобывающие отрасли, автомобилестроение и строительство). Иными словами, это отрасли, в которых руководителям требовалось опираться на эффективное управление IT-отделами ровно настолько, насколько это нужно для достижения их целей[13].

Издержки в человеческих и экономических ресурсах

Когда сотрудники, особенно занимающие невысокую должность в отделе разработки, несколько лет находятся в ловушке нисходящей спирали, они зачастую чувствуют себя заложниками системы, запрограммированной на неудачи, и понимают, что не в силах достичь заметных результатов. Ощущение бессилия нередко сменяется эмоциональным выгоранием, ощущением покорности судьбе, цинизмом и даже безнадежностью и отчаянием.

Психологи утверждают: создание системы, порождающей чувство бессилия, – одна из основных опасностей, которым мы подвергаем своих близких. Мы лишаем других возможности управлять собственными достижениями, создавая культурную среду, где подчиненные опасаются поступать правильно из-за страха наказания, неудачи или риска потерять средства к существованию. В результате формируются условия для появления *приобретенной беспомощности*: инженеры теряют желание или способность поступать так, чтобы избежать подобных проблем в будущем.

Для сотрудников это означает сверхурочные, работу по выходным и ухудшение качества жизни, причем не только их самих, но и всех тех, кто от них зависит, в том числе членов семьи и друзей. Неудивительно, что, когда складывается подобная ситуация, мы теряем лучших (исключая тех, кто обладает гипертрофированным чувством ответственности или связан какими-то обязательствами).

Помимо описанных неудобств, существующие способы действия провоцируют также финансовые потери, а ведь их можно было избежать. Можно оценить такие издержки в 2,6 триллиона долларов в год. На момент написания книги – сумма, равная валовому внутреннему продукту Франции (как считается, шестой экономики в мире).

Предлагаем следующие расчеты: по оценкам компании IDC и компании Gartner, примерно 5 % общемировой суммы валового внутреннего продукта (3,1 триллиона долларов) тратятся на IT-отрасли (оборудование, услуги, телекоммуникации). Если считать, что 50 % этой суммы ушли на операционные расходы и поддержание существующих систем, а треть этих 50 % была потрачена на незапланированные работы и переделку, то впустую было потрачено примерно 520 миллиардов долларов.

Если применение методов DevOps поможет уменьшить потери за счет лучшего управления и повышения качества результата и увеличить потенциал сотрудников в пять раз (и это по самым скромным подсчетам), то можно получить дополнительно 2,6 триллиона долларов в год.

Это мы обсудим ниже. *Прим. авт.*

[13] Например, Вернон Ричардсон с коллегами опубликовал следующие поразительные результаты. Они изучили отчеты 184 публичных корпораций по форме 10-K и разделили их на три группы: а) фирмы с нехваткой материальных ресурсов и с нечеткой работой IT-подразделений; б) фирмы с нехваткой материальных ресурсов и с четкой работой IT-подразделений; в) «чистые фирмы» без нехватки материальных ресурсов. В фирмах из группы А текучка кадров среди руководителей высшего звена была в восемь раз выше, чем в фирмах из группы В, а в фирмах из группы Б – только в четыре раза. Ясно, что работа IT-структур оказывается гораздо более важной, чем принято считать. *Прим. авт.*

Этическая сторона DevOps: лучший путь

В предыдущих разделах мы описали проблемы и отрицательные последствия существующего положения вещей, вызванного хроническим конфликтом, – начиная от неспособности организации достичь целей и заканчивая вредом, причиняемым нам. DevOps решает эти проблемы, одновременно позволяя увеличить производительность организации в целом, обеспечить достижение различными отделами (например, разработчиками, контролем качества, IT-эксплуатацией, отделом информационной безопасности) своих функциональных целей и улучшить условия для сотрудников.

Такое удивительное сочетание наглядно объясняет, почему DevOps вызывает восхищение и энтузиазм и так быстро завоевывает сторонников, включая лидеров в области технологий, инженеров и других участников значительной части экосистемы разработки ПО.

Разрываем нисходящую спираль, используя DevOps

В идеале небольшие команды разработчиков трудятся независимо, разрабатывая функциональности, проверяя их правильность в среде, приближенной к реальной, и быстро, надежно и безопасно развертывая их в производственной среде. Развертывание кода – рутинная и предсказуемая процедура. Вместо того чтобы начинать развертывание поздно вечером в пятницу и тратить все выходные, его можно выполнять в разгар рабочего дня, когда все сотрудники на своих местах. При этом они даже не будут замечать развертывания – за исключением случаев, когда обнаружат появление новых функций или исчезновение ошибок, что приведет их в восторг. К тому же, осуществляя развертывание кода в середине рабочего дня, сотрудники IT-эксплуатации в первый раз за долгие десятилетия получат возможность трудиться как все нормальные люди – в рабочее время!

Благодаря созданию контуров быстрой обратной связи на каждом этапе процесса любой исполнитель имеет возможность сразу же увидеть результат своих действий. Когда изменения зафиксированы в системе контроля версий, быстрые автоматизированные тесты проводятся в среде, близкой к производственной, снова и снова подкрепляя уверенность, что и код, и среда работают как запланировано, всегда безопасны и готовы к развертыванию.

Автоматизированное тестирование дает разработчикам возможность быстро обнаруживать ошибки (обычно за минуту), что позволяет сразу же их исправлять и учиться тому, что невозможно работать нормально, если ошибка обнаруживается спустя шесть месяцев после интеграционного тестирования, когда связи между причиной и эффектом уже давно выветрились из головы. Вместо того чтобы накапливать технический долг, следует устранять проблемы сразу после обнаружения, если необходимо – с привлечением всей организации, поскольку ее глобальные цели перевешивают локальные цели группы или даже отдела.

Всеобъемлющий сбор телеметрии о коде и программной среде обеспечивает своевременное обнаружение проблем и их быстрое исправление. Он подтверждает: все на месте, как предусмотрено, и клиенты получают продукт благодаря предоставленному нами ПО.

При таком сценарии каждый чувствует себя на своем месте: архитектура процесса дает возможность небольшим командам действовать уверенно, даже не будучи тесно привязанными к работе, выполняемой другими командами, и использовать платформы с самообслуживанием, повышающие коллективный опыт отделов IT-эксплуатации и информационной безопасности. Вместо того чтобы постоянно ждать результатов от смежных групп, а потом переделывать заново огромный объем работы, команды трудятся независимо и продуктивно над небольшими заданиями, быстро и часто предоставляя клиентам новые возможности.

Даже релиз широко известных, привлекающих внимание продуктов становится обычным делом, если используются методы «теневого запуска». Задолго до даты запуска код помещается в рабочую среду, но его новые возможности невидимы для всех, кроме персонала компании-разработчика и небольшой группы реальных клиентов, что позволяет тестировать и развивать новые возможности, пока не будет достигнута поставленная бизнес-цель.

И вместо того чтобы трудиться дни и ночи напролет, пытаясь задействовать новую функциональность, мы просто включаем ее в конфигурационных настройках. Это небольшое изменение делает ее доступной огромному количеству клиентов и предоставляет возможность вернуться назад, если что-либо пойдет не так. В результате релиз нового продукта полностью контролируется, он предсказуем, обратим и не вызывает опасений.

Спокойнее проходят не только те релизы, где добавляются новые функции. На ранних стадиях обнаруживаются и исправляются любые проблемы, пока они не разрослись до катастрофических размеров. Их исправление окажется быстрее и дешевле. С каждым новым исправлением мы добываем для организации новое знание, что позволяет успешно предотвращать проблемы, а впоследствии быстрее обнаруживать и исправлять схожие.

Кроме того, инженеры постоянно обучаются, и при этом формируется организационная культура, основанная на выдвижении гипотез. Научный метод применяется для того, чтобы все подвергалось оценке, а разработка продукта и улучшение процесса разработки расцениваются как эксперименты.

Мы ценим время каждого сотрудника, поэтому не тратим годы на создание функций, не нужных клиентам, не развертываем неработающий код и не исправляем то, что не вызывает проблем.

Будучи нацелены на решение поставленных задач, мы создаем долговременные группы, и они полностью отвечают за достижение нужных результатов. Завершив какой-либо проект, мы не перераспределяем разработчиков по другим командам, в результате чего они лишаются возможности следить за результатами своего труда. Мы сохраняем состав команд. Поэтому их члены могут повторять итерации разработки, улучшая продукт и используя накопившиеся знания, чтобы успешнее достигать поставленных целей. То же самое относится и к командам, создающим продукт для внешних клиентов, так же как внутренние платформенные команды помогают другим командам действовать продуктивнее, успешнее и безопаснее.

Вместо культуры страха мы создаем культуру высокого доверия и сотрудничества. Никто не боится брать на себя риски. Все получают возможность смело обсуждать любые проблемы, не скрывая их и не откладывая на потом. В конце концов, чтобы решить проблему, ее нужно распознать.

И поскольку каждый сам определяет качество своей работы, то он и занимается встраиванием автоматизированного тестирования в свою повседневную деятельность и использует партнерские проверки, чтобы быть уверенным: проблемы будут устранены задолго до того, как окажут воздействие на клиента. Эти процессы снижают риски по сравнению с принятой практикой утверждения кода руководителями, что позволяет предоставлять продукт быстро, надежно и безопасно и даже убеждать скептически настроенных аудиторов, что у нас есть эффективная система внутреннего контроля.

И если что-то вдруг пойдет не так, мы тщательно анализируем причины неудачи – не чтобы наказать виновного, а чтобы лучше понять, чем вызван сбой и как предотвращать подобное в будущем. Такой ритуал укрепляет культуру обучения. У нас существуют также внутренние конференции, и на них работники могут улучшить навыки: каждый постоянно учит других и учится сам.

Поскольку мы заботимся о качестве, постольку иногда специально создаем сбои в производственной среде, чтобы понять, каким образом система выходит из строя в случаях, которые можно предвидеть. Мы проводим запланированные тренировки по устранению круп-

номасштабных сбоев, случайным образом «убивая» процессы в производственной системе или выключая серверы, вводим сетевые задержки и делаем другие злонамеренные поступки, чтобы проверить отказоустойчивость. Это позволяет повысить устойчивость системы к сбоям, а также провести обучение персонала и внести улучшения по результатам тестов.

В таком мире каждый человек, какова бы ни была его роль в технологической организации, – хозяин своего труда. Он уверен, что его работа действительно способствует достижению целей компании, что подтверждается низким уровнем сбоев рабочей среды и успехом организации на рынке.

Ценность DevOps для бизнеса

У нас немало убедительных доказательств ценности методов DevOps для бизнеса. В «Докладе о состоянии *DevOps*» компании Puppet Labs (в создании участвовали Джез Хамбл и Джин Ким) представлены данные за 2013–2016 гг., полученные примерно от 25 000 технических специалистов. Это помогает лучше понять уровень работоспособности и особенности организаций на всех этапах внедрения DevOps.

Прежде всего, эти данные продемонстрировали высокую (по сравнению с прочими) эффективность компаний, использующих DevOps. Сравнивались следующие характеристики:

- показатели пропускной способности;
- развертывание кода и внесение изменений (чаще в 30 раз);
- время, необходимое на разработку кода и внесение изменений в него (в 200 раз меньше);
- показатели надежности;
- развертывание в производство (коэффициент успешности в 60 раз выше);
- среднее время восстановления после сбоя (в 168 раз быстрее);
- показатели эффективности организации;
- производительность, доля на рынке и рентабельность (вероятность улучшить эти показатели в два раза выше);
- рост капитализации рыночной стоимости компании (за три года на 50 % больше).

Другими словами, более производительные компании оказались и более гибкими, и более надежными. Это наглядное доказательство того, что использование DevOps позволяет выйти из корневого, хронического конфликта. Организации с высокой эффективностью развертывали код в 30 раз чаще, а время, необходимое для перехода от «код зафиксирован» к «успешно работает в реальной среде», было меньше в 200 раз – с периода в несколько недель, месяцев (а порой и квартала) оно сократилось до нескольких минут или часов.

Кроме того, у более производительных компаний имелось в два раза больше шансов повысить прибыльность, долю на рынке и эффективность. А у тех, кто сообщил нам свои биржевые символы, рост капитализации за три года оказался выше на 50 %. Текучка кадров в них ниже, а сотрудники чувствуют удовлетворенность и в 2,2 раза чаще рекомендуют компанию друзьям как отличное место работы[14]. Организации с высокой эффективностью лучше обеспечивают такой показатель, как «информационная безопасность». Достижение целей по

[14] По данным индекса чистой лояльности сотрудников (employee Net Promoter Score – eNPS). Это очень значимое открытие, поскольку исследование доказало, что «компании, где работники имеют высокую удовлетворенность, показали в 2,5 раза более высокий рост доходов, чем компании с низкой удовлетворенностью работников. И [торгующиеся на бирже акции] компании с высоким уровнем удовлетворенности показали за период с 1997 по 2011 г. рост цены, втрое превышающий рост биржевых индексов». *Прим. авт.*

безопасности интегрировано во все стадии процессов разработки и эксплуатации, поэтому на исправление соответствующих проблем уходит вдвое меньше времени.

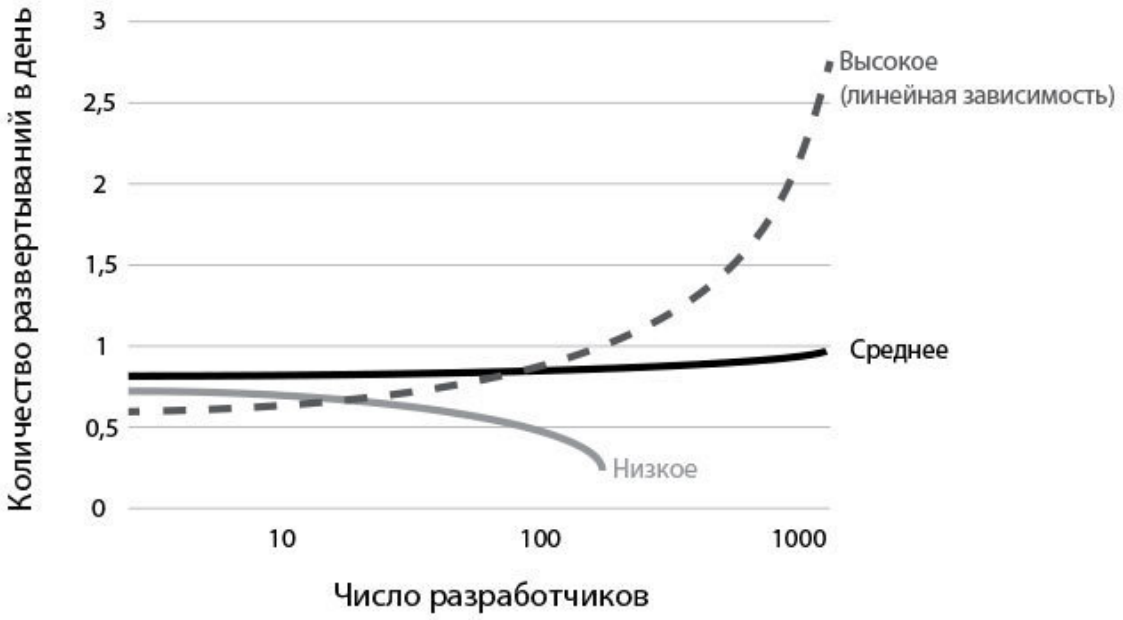
DevOps помогает увеличивать продуктивность разработчиков

Когда число разработчиков увеличивается, производительность каждого нередко снижается из-за потери времени на коммуникации, интеграцию и избыточное тестирование. Этот феномен подробно описан в книге Фредерика Брукса «Мифический человеко-месяц, или Как создаются программные системы»[15]. В ней он объясняет: когда работа над проектом не выполняется в срок, увеличение количества разработчиков снижает продуктивность не только каждого в отдельности, но и команды в целом.

С другой стороны, DevOps показывает, что при наличии правильной архитектуры проекта, правильных технических методов и правильной производственной культуры небольшие команды разработчиков способны быстро, надежно и независимо от других разрабатывать, выполнять интеграцию, тестировать и развертывать изменения в производственной среде. Согласно наблюдениям Рэнди Шупа, бывшего директора по разработке в компании Google, крупные компании, использующие DevOps, имеют в штате тысячи разработчиков, но их организация и методы позволяют небольшим командам добиваться удивительной продуктивности, как если бы они были стартапом.

В «Докладе о состоянии DevOps в 2015 г.» изучался не только показатель «количество развертываний в день», но и «количество развертываний в день на одного разработчика». Мы выдвинули гипотезу, что высокопроизводительные инженеры могут увеличивать количество развертываний по мере роста численности команды.

Действительно, мы обнаружили такую зависимость. На рис. 1 показано, что при низкой производительности разработчиков количество развертываний в день уменьшается по мере роста численности команды, при средней – остается постоянным, а при высокой – растет пропорционально числу разработчиков.



[15] М.: Символ-Плюс, 2010. *Прим. перев.*

Рис. 1. Количество развертываний в день в зависимости от числа разработчиков (источник: «Доклад о состоянии DevOps в 2015 г.», компания Puppet Labs)[16]

Другими словами, организации, внедрившие DevOps, могут увеличивать количество развертываний в день пропорциональным увеличением числа разработчиков, как это уже сделали компании Google, Amazon и Netflix[17].

Универсальность решения

Одна из наиболее авторитетных книг по бережливому производству – «Цель. Процесс непрерывного совершенствования»[18] – написана доктором Элияху Голдраттом в 1984 г. Под ее влияние попало целое поколение руководителей предприятий во всем мире. Это рассказ о директоре завода. Он должен был сократить расходы и наладить выполнение поставок всего за 90 дней, потому что иначе предприятие пришлось бы закрывать.

Позже Голдратт рассказал о письмах, полученных после выхода книги. Обычно их содержание было примерно таким: «Совершенно очевидно, что вы тайно проникли на наш завод, поскольку абсолютно верно описали мою жизнь как директора…» Но гораздо важнее другое: письма показали, что люди способны повторить прорывные действия у себя в организации.

Книга Джина Кима, Кевина Бера и Джорджа Слаффорда «Проект “Феникс”. Роман о том, как DevOps меняет бизнес к лучшему» (год выпуска – 2013-й) очень похожа на «Цель…». Это повесть о руководителе IT-организации, столкнувшемся с проблемами, типичными для такого рода компаний: перерасход бюджета, нарушение графика, хотя его соблюдение жизненно важно для компании. Развертывание проекта происходит из рук вон плохо, возникают сложности с доступностью, безопасностью, соответствием требованиям и так далее. В конечном счете он и его команда начинают использовать методы и принципы DevOps, чтобы справиться с этими проблемами и дать организации возможность выдержать конкуренцию на рынке. Помимо этого, показано, как методы DevOps улучшают рабочую атмосферу в команде, способствуют снижению стресса, повышают удовлетворенность результатами вследствие большей вовлеченности сотрудников в процессы организации в целом.

Как и в случае с «Целью…», в «Проекте “Феникс”…» имеются серьезные доказательства универсальности описанных в книге проблем и решений. Приведем некоторые отзывы, взятые с сайта Amazon: «Я обнаружил сходство между собой и героями этой книги… Вполне возможно, что я встречал таких людей на всем протяжении моей карьеры». «Если вам доводилось работать в любом качестве в IT, эксплуатации или в сфере информационной безопасности, вы, безусловно, почувствуете связь с описанным в этой книге». «В книге “Проект «Феникс»…” нет такого персонажа, которого я не смог бы сравнить с собой или с кем-то, кого знаю в реальной жизни… не говоря уж о проблемах этих персонажей».

В остальных частях этой книги мы опишем, как повторить преобразования, описанные в «Проекте “Феникс”…», и приведем примеры из жизни других организаций, использующих методы и принципы DevOps, чтобы добиться такого же успеха.

---

[16] Показаны данные только по тем организациям, которые выполняют хотя бы одно развертывание в день. *Прим. авт.*

[17] Другой выдающийся пример – компания Amazon. В 2011 г. она выполняла около 7000 развертываний в день. В 2015 г. – уже 130 000. *Прим. авт.*

[18] Издана: М.: Альпина Диджитал, 2014. *Прим. перев.*

Руководство по DevOps: краткий обзор

Цель данной книги в том, чтобы описать теорию, принципы и методы, необходимые для успешного использования DevOps и достижения желаемых результатов.

В основу этого руководства положены десятилетия изучения теории рационального использования высокопроизводительных технологических организаций. Мы проделали эту работу, помогая компаниям преобразовываться. Наши исследования подтвердили эффективность методов DevOps. Также были использованы материалы бесед с экспертами в соответствующих областях и анализ около ста практических примеров, о которых шла речь на конференции DevOps Enterprise Summit.

Книга разделена на шесть частей, описывающих теорию DevOps, принципы использования «трех путей», особый взгляд на обоснование теории, изложенной в книге «Проект “Феникс”. Роман о том, как DevOps меняет бизнес к лучшему» и предназначенной для каждого, кто когда-то занимался технологическими процессами создания продуктов (обычно это понятие включает управление, разработку, тестирование, эксплуатацию и информационную безопасность) или был иным образом вовлечен в указанные процессы. Также книга предназначена для руководителей бизнеса и маркетологов, ведь именно в этих областях обычно зарождаются технологические инициативы.

Не следует ожидать, что в книге есть глубокий анализ любого из этих направлений, а также DevOps, Agile, ITIL, принципов бережливого управления или описания улучшения процессов. Но каждая из тем затрагивается и разъясняется в других специализированных изданиях, если требуется.

Наша цель – сформировать базовые знания о важнейших концепциях в каждой из областей. Это нужно, чтобы понять основы, а также ввести терминологию и формулировки, необходимые на практике работы для взаимодействия с коллегами, вовлеченными в процесс создания IT-ценностей. Ну и для того, чтобы сформулировать общие цели.

Книга окажется полезной руководителям бизнеса и другим заинтересованным лицам, если они полагаются на технологические организации.

Кроме того, книга предназначена тем, кто в своих организациях лишь отчасти сталкивается с описанными здесь проблемами (например, с длительным временем развертывания или со сложным развертыванием, сопряженным с неприятностями). Такие читатели, оказавшиеся в относительно удачном положении, смогут извлечь пользу из понимания принципов DevOps, особенно касающихся общих целей, обратной связи и непрерывного обучения.

В части I приведен краткий обзор истории DevOps и представлены основы теории и ключевые темы из соответствующих областей знаний, развивавшихся несколько десятилетий. Затем мы представим принципы верхнего уровня – подход «трех путей»: обеспечение полного цикла разработки, петля обратной связи, культура непрерывного обучения и экспериментирования.

В части II описывается, когда и как надо начинать, представлены такие концепции, как поток создания ценности, принципы и шаблоны организационного проектирования, адаптационные шаблоны, а также приводятся практические примеры.

В части III описывается, как ускорить процесс разработки через формирование основы для конвейерного развертывания – быстрого и эффективного автоматизированного тестирования, непрерывной интеграции, непрерывной поставки и проектирования низкорискового процесса релизов.

В части IV обсуждаются способы ускорения и усиления обратной связи путем создания эффективной телеметрии для обнаружения, более глубокого анализа и решения проблем, достижения поставленных целей, формирования обратной связи, позволяющей разработке и

эксплуатации безопасно развертывать изменения, встраивать A/B-тестирование в повседневную деятельность и создавать процессы взаимной проверки и координации для повышения качества результатов.

В части V описывается, как ускорять непрерывное обучение, формируя соответствующую внутреннюю культуру, преобразуя небольшие открытия в глобальные улучшения и правильно резервируя время для организационного обучения и усовершенствований.

И наконец, в части VI мы расскажем, как правильно обеспечивать безопасность и проверять соответствие ее требованиям в повседневной работе, интегрируя превентивный контроль безопасности в общие хранилища исходного кода и услуг, функции обеспечения безопасности – в конвейер развертывания. Необходимо улучшать телеметрию для обеспечения более успешного обнаружения ошибок и восстановления, защиты конвейера развертывания и достижения целей по изменению управления.

Вводя кодификацию этих приемов, мы надеемся ускорить внедрение методов DevOps, повысить их успешность и уменьшить количество энергии, необходимой для запуска преобразований DevOps.

Часть I. «Три пути»

Введение

В первой части книги «Руководство по DevOps» мы рассмотрим, как слияние нескольких важных тенденций в менеджменте и технологиях создает предпосылки для появления на сцене DevOps. Мы опишем потоки создания ценностей, то, как DevOps становится результатом применения принципов бережливого производства к потокам создания технологических ценностей, а также «три пути»: потоки, обратная связь и постоянное обучение и экспериментирование.

Основные темы части I:

• принцип потока ценности, ускоряющий для клиентов доставку продукта от разработчиков к отделу эксплуатации;

• принцип обратной связи, позволяющий создавать безопасные системы;

• принцип постоянного обучения и экспериментирования, создающий режим наибольшего благоприятствования для организации высокой культуры производства при научном подходе к рискам как составляющей ежедневной работы.

Краткая история

DevOps и связанные с ним технические, архитектурные и культурные методики – результат соединения многих философских и управленческих тенденций. Во многих организациях смогли разработать сходные принципы самостоятельно. Понимание того, что DevOps появился из широкого спектра таких действий, феномен, описанный Джоном Уиллисом (одним из соавторов этой книги) как «конвергенция DevOps», показывает удивительный прогресс мышления и невероятные совпадения. Опыт длиной в несколько десятилетий, полученный при совершенствовании производства, организации управления, простроенного на высоком уровне доверия, и прочего, привел к тому, что мы сейчас называем методами DevOps.

DevOps – результат применения самых надежных принципов из области лидерства и производства материальных ценностей к потоку создания ценности IT. DevOps опирается на базовые понятия теории бережливого производства, теории ограничений, производственной системы компании Toyota, правила создания устойчивых систем, модель обучающихся организаций, культуру психологической безопасности и так далее. Другие важные особенности, которые используют в DevOps, – это управленческая культура высокого доверия, лидерство как служение и управление изменениями в организации. В результате стали возможными высочайшее качество, надежность, стабильность и безопасность, полученные меньшей кровью и за меньшие деньги, увеличение роста и стабильности за счет потока технологической ценности в подразделениях управления продуктом, разработки, тестирования, отделах управления IT-эксплуатацией и информационной безопасности.

Хотя можно сказать, что DevOps основывается на принципах бережливого производства, теории ограничений и распространенного в Toyota движения «ката», многие рассматривают его как логическое продолжение Agile-движения, зародившегося в 2001 г.

Принципы бережливого производства

Такие техники, как «управление созданием потока ценности» и «канбан-доски», были созданы в рамках производственной системы Toyota в 1980-х гг. В 1997 г. Lean Enterprise Institute начал изучать возможность применения принципов бережливого производства к другим вариантам потоков создания ценности, например к сфере обслуживания и здравоохранению.

Два основных принципа бережливого производства заключаются в том, что по *продолжительности производственного цикла*, необходимого для превращения сырья в готовый продукт, можно максимально точно предсказать качество продукта, степень удовлетворенности клиентов и настроение исполнителей. Лучший способ сделать производственный цикл короче – сократить размер каждого производственного задания.

Принципы бережливого производства сфокусированы на том, как с помощью системного мышления создать нечто ценное для клиента за счет постоянства целей, развития научного подхода, потока создания ценности, методов «вытягивания» (pull) (против «заталкивания» (push)), обеспечения высокого качества с первого раза и высокой культуры руководства.

Agile-манифест

В 2001 г. появился манифест Agile. Его создали 17 авторитетных специалистов в области разработки ПО. Они хотели создать легковесный набор ценностей и принципов в противовес тяжеловесным процессам разработки (например, метод водопада) и методологиям (например, так называемый RUR – рациональный унифицированный процесс).

Один из ключевых принципов манифеста – «эффективно доставлять программное обеспечение часто, раз в две недели или раз в два месяца, делая упор на сокращение сроков». Подчеркивалось стремление к небольшому объему производственных заданий и релизу продуктов с инкрементальными изменениями вместо крупных, «каскадных». Другие принципы подчеркивали необходимость в небольших самомотивированных командах, действующих в атмосфере высокого уровня доверия.

Считается, что методы Agile способны значительно увеличить продуктивность организаций-разработчиков. И, что интересно, многие из ключевых моментов в истории DevOps соответствуют аналогичным ситуациям, происходившим в сообществе Agile или на конференциях, посвященных этому методу.

Agile-инфраструктура и движение Velocity

В 2008 г. в рамках конференции Agile, проходившей в Торонто, Патрик Дюбуа и Эндрю Шафер провели встречу под названием «Птицы одного полета». Она была посвящена применению принципов Agile к инфраструктуре, а не написанию кода приложений. Поначалу они были единственными приверженцами этой идеи, но быстро обрели единомышленников, включая одного из авторов этой книги – Джона Уиллиса.

Позже, на конференции Velocity в 2009 г., Джон Оллспоу и Пол Хаммонд продемонстрировали новаторскую презентацию под названием «Десять развертываний в день: кооперация разработки и эксплуатации во Flickr». В ней они описали, как создали общие цели для разработчиков (Dev) и эксплуатация (Ops) и использовали методы непрерывной интеграции, чтобы сделать развертывание частью ежедневной работы. Как утверждали впоследствии участники

презентации, они сразу же поняли, что присутствуют на историческом мероприятии, имеющем далеко идущие последствия.

Патрик Дюбуа не участвовал в презентации, но был настолько восхищен идеей Оллспоу и Хаммонда, что в том же 2009 г. организовал первую конференцию DevOpsDays в бельгийском Генте, где тогда жил. Так и появился на свет термин DevOps.

Непрерывная поставка

Внедряя непрерывную разработку, тестирование и интеграцию, Джез Хамбл и Дэвид Фарлей расширили концепцию *непрерывной поставки.* Это определило важность «конвейера разработки»: код и инфраструктура всегда готовы к развертыванию, весь код прошел проверку и может быть безопасно развернут в производственной среде. Идею впервые представили на конференции Agile в 2006 г. Затем, в 2009 г., Тим Фитц абсолютно независимо придумал то же самое и изложил свои мысли в блоге под заголовком «Непрерывное развертывание» [19].

Ката компании Toyota

В 2009 г. Марк Ротер написал книгу «Тойота Ката. Лидерство, менеджмент и развитие сотрудников для достижения выдающихся результатов» [20], где изложил двадцатилетний опыт кодифицирования производственной системы Toyota. Будучи студентом, он принял участие в поездке руководителей компании General Motors на заводы компании Toyota. Затем ему довелось участвовать в разработке инструментария для внедрения бережливого производства. Он был очень удивлен, что ни одна из компаний, внедривших этот метод, не смогла достичь такого же уровня производительности, как Toyota.

Марк сделал вывод: сообщество Lean упустило из вида наиболее важную часть метода – *улучшение ката*[21]. Он пояснил: каждая организация имеет свои рабочие процедуры, и улучшение ката требует создания структуры для ежедневного, рутинного совершенствования, поскольку повседневный опыт – это и есть то, что повышает результаты. Постоянно действующий цикл, то есть желаемое будущего состояния, еженедельное формулирование целей и непрерывное совершенствование повседневной работы, и составлял основу улучшений в компании Toyota.

Выше мы описали историю DevOps и связанных с ним течений. На их основе и возник метод DevOps. Далее в первой части мы рассмотрим потоки создания ценности, то, как к потокам технологической ценности можно применить принципы бережливого управления и «трех путей», обратной связи и непрерывного обучения и экспериментирования.

---

[19] DevOps также расширяет и усовершенствует принцип инфраструктура как код, впервые предложенный Марком Берджессом, Люком Канисом и Адамом Джекобом. При использовании этого принципа работа эксплуатации автоматизирована и трактуется как разработка кода приложений, так что все современные методики разработки могут быть применены ко всему потоку разработки и управления. Это увеличивает возможности ускорения потока разработки, включая непрерывную интеграцию (придумана Греди Бучем как один из 12 ключевых методов «экстремального программирования»), непрерывную поставку (впервые введена Джезом Хамблом и Дэвидом Фарли) и непрерывное развертывание (созданное компаниями Etsy и Wealthfront, а также Эриком Рисом из IMVU). *Прим. авт.*

[20] Издана: СПб.: Питер, 2014. *Прим. перев.*

[21] Ката – формализованная последовательность движений, связанных принципами ведения поединка с воображаемым противником или группой противников. По сути, является квинтэссенцией техники конкретного стиля боевых искусств. *Прим. ред.*

Глава 1. Agile, непрерывная поставка и «три пути»

Здесь представлено введение в основы теории бережливого производства и «трех путей» – из этих принципов сформировалось современное состояние DevOps.

Внимание уделено в первую очередь теории и принципам, выработанным за несколько десятилетий изучения рабочих сред, организации его высокой надежности, создания моделей управления с высоким уровнем доверия, положенных в основу методов DevOps. Получившиеся в результате принципы и схемы действий, а также их практическое применение для создания потока технологической ценности описаны в остальных главах книги.

Производственный поток ценности

Одна из фундаментальных концепций бережливого производства – *поток создания ценности*. Сначала мы определим это понятие в рамках промышленности, а затем экстраполируем на DevOps и технологический поток создания ценности.

Карен Мартин и Майк Остерлинг в книге Value Stream Mapping: How to Visualize Work and Align Leadership for Organizational Transformation определили поток создания ценности как «последовательность действий, предпринимаемых организацией с целью выполнить запрос клиента», или как «последовательность действий, необходимых для разработки, выпуска и доставки товаров клиенту, включая оба потока – и информационный, и материальный».

В материальном производстве поток создания ценности легко увидеть, легко наблюдать за ним: он начинается, когда получен заказ от клиента, а сырье для производства поступило на склад. Чтобы обеспечить короткое и предсказуемое время выполнения заказа в любом потоке создания ценности, обычно фокусируются на плавном течении работы, используя такие приемы, как уменьшение размера партии, снижение объема незавершенного производства, недопущение переделок, чтобы исключить попадание дефектных деталей на конечных этапах и постоянно оптимизировать систему для движения в сторону глобальных целей.

Технологический поток ценности

Те же принципы и методы, обеспечивающие скорость выполнения работы в процессах материального производства, применимы и к области технологий (и вообще для любой деятельности, создающей знание). В DevOps поток создания технологической ценности обычно определяют как процесс, требующийся для преобразования бизнес-гипотез в технологический сервис, поставляющий ценность заказчику.

Исходные данные для процесса – формулирование бизнес-цели, концепции, идеи или гипотезы. Все начинается, когда мы принимаем задачу в области разработки, добавляя ее к уже имеющимся.

Таким образом, команда разработчиков, следующая типичному Agile- или итеративному процессу, скорее всего, сможет адаптировать эту идею в соответствии с пожеланиями пользователей, создав некоторую разновидность спецификации функциональных возможностей. Затем она реализует их в виде кода приложения или создаваемого сервиса и поместит его в репозиторий системы контроля версий, где каждое изменение интегрируется в основной код и тестируется вместе со всей системой.

Поскольку продукт создается, когда наши сервисы работают в производственной среде, следует обеспечить не только быстрый поток доставки, но и то, чтобы развертывание можно

было выполнять без хаоса и перебоев: задержек в обслуживании, нарушения функционирования сервисов, требований безопасности или совместимости.

Фокус на времени развертывания

В остальной части книги сосредоточимся на такой составной части потока создания ценности, как сокращенное время развертывания. Начинается оно тогда, когда некий инженер[22] из команды потока создания ценностей (включающей разработчиков, тестировщиков, отделы эксплуатации и информационной безопасности) получает изменения от системы контроля версий. А заканчивается, когда изменение начинает успешно работать в производственной среде, создавая продукт для клиентов и генерируя обратную связь и телеметрию.

Первый этап включает проектирование и разработку. Он сродни бережливой разработке продуктов и характеризуется высокой изменчивостью и неопределенностью, часто требует творческого подхода к выполнению работы, которая может никогда не понадобиться. Это приводит к различной продолжительности данного этапа. В отличие от него второй этап, включающий тестирование и производство, соответствует принципам бережливого производства. Он требует творческого подхода и опыта и имеет тенденцию к предсказуемости и автоматизации, его цель – обеспечить полезный результат, то есть небольшое предсказуемое время выполнения и близкое к нулю количество дефектов.

Мы не любители больших порций работы, последовательно проходящих через такие части потока создания ценности, как «проектирование и разработка», а затем через поток «тестирование и производство» (например, когда используется большой водопадный процесс или долго живущая функциональная ветвь кода). Напротив, мы стремимся, чтобы тестирование и производство выполнялись одновременно с проектированием и разработкой. Так обеспечиваются быстрый ход потока и высокое качество. Этот метод успешен, если исполнение осуществляется по небольшим частям и качество обеспечивается на каждом этапе потока создания ценностей[23].

Определения: «время выполнения заказа» и «время производства»

В Lean время выполнения заказа – одна из двух характеристик, обычно использующихся для измерения производительности потоков ценности. Другая характеристика – время производства (иногда его еще называют временем контактирования или временем выполнения задачи)[24].

Если отсчет времени выполнения заказа начинается в момент оформления и заканчивается при выполнении, то время производства отсчитывается с момента, когда мы начинаем работу над заказом, точнее, не засчитывается тот период времени, когда заказ стоял в очереди на выполнение (рис. 2).

---

[22] Здесь и далее слово «инженер» означает любого человека, работающего в команде, производящей поток ценности, а не только разработчиков. *Прим. авт.*

[23] По правде говоря, при использовании таких методик, как разработка через тестирование, тестирование может начинаться еще до того, как будет написана первая строка кода. *Прим. авт.*

[24] В этой книге термин «время производства» будет использоваться по той же причине, о которой говорили Карен Мартин и Майк Остерлинг: «чтобы минимизировать возможную путаницу, мы избегаем использовать термин “время цикла”, поскольку он имеет несколько значений, в том числе синонимичное для времени производства и темпа или частоты выдачи результатов, и это только некоторые». *Прим. авт.*

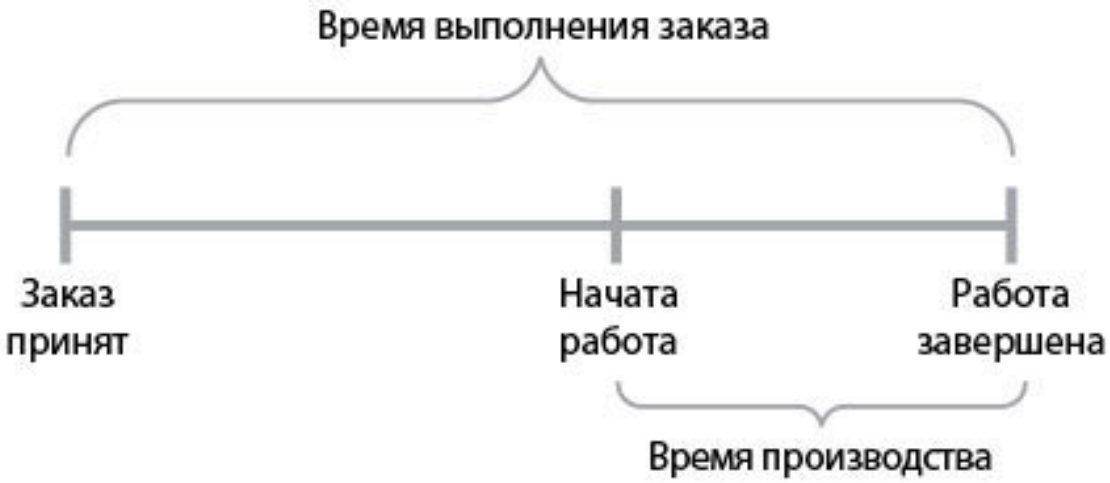


Рис. 2. Время выполнения заказа и время производства

Поскольку время выполнения заказа – самая важная характеристика для клиента, обычная цель – улучшить процессы, сократив вот этот параметр, а не время производства. Однако соотношение времени производства и времени выполнения заказа – важный критерий оценки эффективности: чтобы обеспечить быстрый ход работ и короткое время выполнения заказа, почти всегда требуется сократить время ожидания, пока дойдет очередь.

Обычный сценарий: развертывание требует месяцев

При ведении бизнеса обычными способами мы часто оказываемся в ситуации, когда развертывание занимает несколько месяцев. Особенно часто это происходит в больших сложных организациях, использующих тесно связанные друг с другом монолитные приложения, плохо интегрированные в среду для тестирования, со значительной продолжительностью тестирования и длительным временем развертывания в рабочей среде, высокой зависимостью от тестирования вручную и необходимостью одобрения многочисленными инстанциями в компании. Когда такое происходит, поток создания ценности начинает выглядеть, как показано на рис. 3.

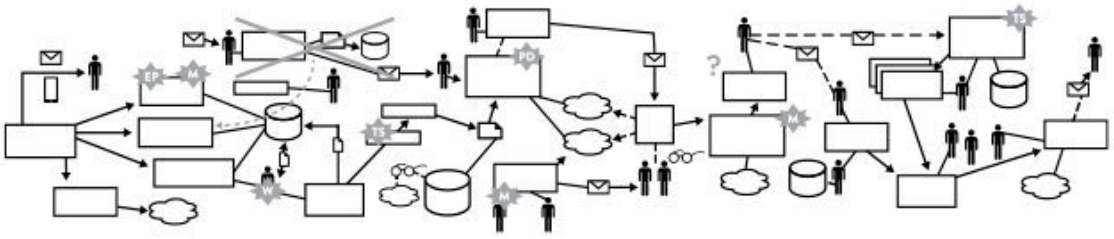


Рис. 3. Поток технологической ценности при продолжительности внедрения длиной в три месяца (пример взят из вышедшей в 2015 г. книги Дэмона Эдвардса DevOps Kaizen)

При большой продолжительности развертывания сверхусилия требуются практически на каждой стадии потока создания ценности. Может оказаться, что при завершении проекта, когда все результаты труда инженеров собраны воедино, все перестанет работать, код не будет собираться правильно или перестанет проходить тесты. Исправление любой проблемы, определение, кто именно «сломал» код и как это исправить, потребует нескольких дней или даже недель, а в результате отдача для клиентов окажется невысокой.

Идеал DevOps – развертывание за минуты

В идеальном случае при работе с DevOps разработчики постоянно быстро получают обратную связь, что дает им возможность быстро и независимо внедрять, интегрировать и валидировать код, а также обеспечивать развертывание кода в производственной среде (это могут делать как они сами, так и другой отдел).

Это достигается за счет постоянной проверки небольших изменений в коде, производимых в репозитории системы контроля версий, выполнения автоматического и пред-производственного тестирования изменений, а затем развертывания в реальной производственной среде. Что и позволяет нам быть твердо уверенными: сделанные изменения после развертывания будут функционировать так, как задумано, и любая возникшая проблема будет быстро обнаружена и исправлена.

Наиболее легко это достигается, когда архитектура модульная, хорошо инкапсулированная, в ней отсутствуют тесные связи между компонентами, так что небольшие группы имеют возможность работать с высокой степенью автономности, возникающие сбои оказываются небольшими и с ограниченными последствиями, не вызывающими глобальных нарушений работоспособности системы.

При таком сценарии время развертывания измеряется минутами или в худшем случае часами. Получившаяся карта потока ценности должна выглядеть примерно так, как показано на рис. 4.

Рис. 4. Технологический поток создания ценности с развертыванием за минуты

Наблюдать за %C/A в качестве оценки необходимости доработок

Помимо времени выполнения заказа и времени производства для оценки технологического потока создания ценности применяется и третий показатель – доля завершенной и правильной работы (percent complete and accurate – %C/A). Этот показатель отражает качество выполнения на каждом этапе потока создания ценности. Карен Мартин и Майк Остерлинг утверждают: «показатель %C/A можно получить, опросив клиентов, как часто в количественном выражении они получали продукт, пригодный к использованию *сразу*. Подразумевается, что они используют результат без необходимости его корректировать, добавлять пропущенную информацию или пояснения к имеющейся, если она недостаточно понятна».

«Три пути»: принципы, лежащие в основе DevOps

В книге The Phoenix Project «три пути» представлены как основополагающие принципы. Из них выводится DevOps и его методы (рис. 5).

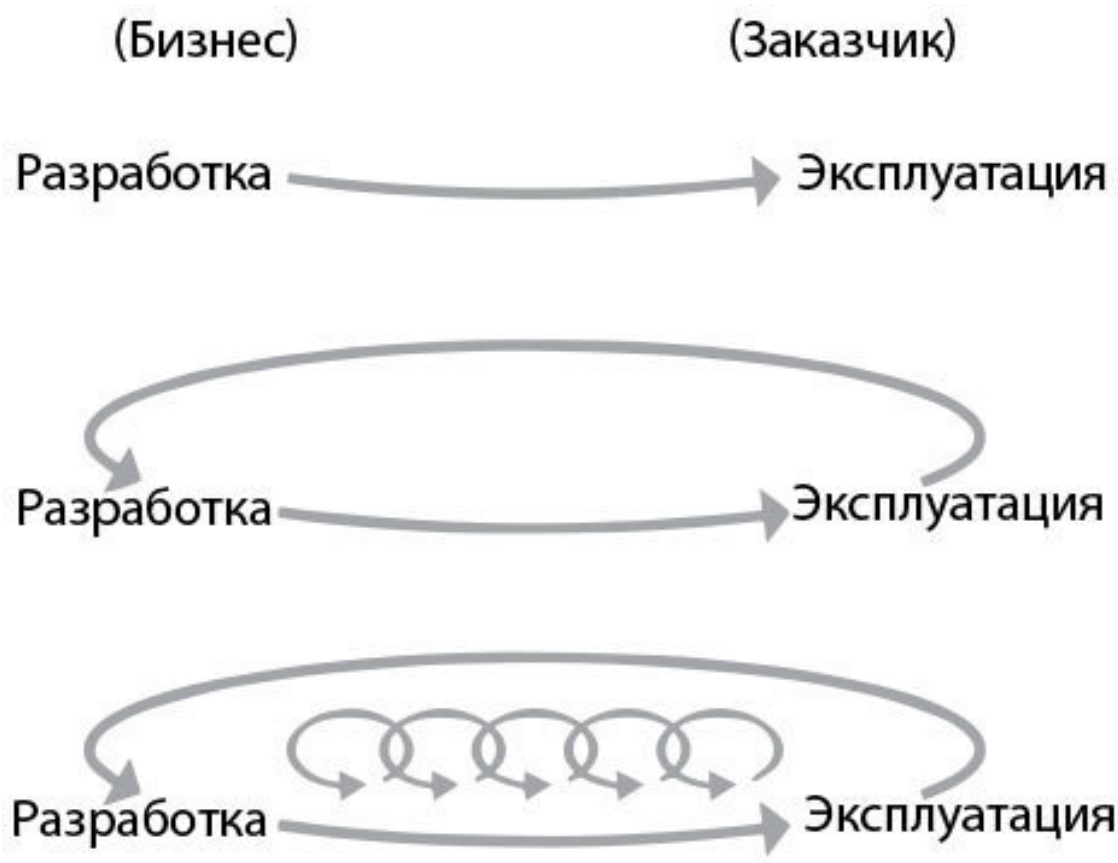


Рис. 5. «Три пути» (пример взят из текста Джина Кима The Three Ways: The Principles Underpinning DevOps, размещенного в блоге Revolution Press blog по адресу: http://itrevolution.com/the-three-ways-principles-underpinning-devops/, доступ осуществлен 9 августа 2016 г.)

Первый путь обеспечивает быстрое течение потока слева направо от разработки к эксплуатации, а затем к клиентам. Чтобы сделать это течение как можно более интенсивным, необходимо сделать результаты видимыми, уменьшая размеры заданий и интервалы между ними, обеспечивая качество путем предотвращения попадания дефектов на конечные этапы и постоянно занимаясь оптимизацией для достижения глобальной цели организации.

За счет ускорения прохождения работы через поток создания технологической ценности мы сокращаем время для выполнения запросов внутренних или внешних клиентов, особенно необходимое для развертывания кода в производственной среде. При этом повышается качество и результативность, а также способность превзойти конкурентов.

Получившиеся в результате методы включают непрерывную разработку, интеграцию и тестирование, а также процессы развертывания: создание различных сред по требованию, ограничение параллельно исполняемых задач и построение систем и организаций, допускающих изменения без риска.

«Второй путь» обеспечивает быстрый и постоянный поток обратной связи справа налево на всех этапах потока создания ценности. Это требует от нас усиления обратной связи с целью предотвратить повторное появление проблем и получить более быстрое обнаружение и восстановление. При этом мы обеспечиваем качество уже на исходном этапе и создаем или встраиваем знание в те этапы, где оно необходимо, – это позволяет нам создавать все более безопасные системы: проблемы обнаруживаются и исправляются задолго до того, как может произойти катастрофический сбой.

Если распознавать проблемы сразу после их появления и решать их, то тем самым можно постоянно укорачивать петли обратной связи, а это один из основных постулатов всех без исключения современных методологий совершенствования процессов. Это делает максимально перспективными возможности нашей организации учиться и улучшаться.

Третий путь позволяет создать продуктивную культуру высокого доверия. Она поддерживает динамический, упорядоченный и научный подход к экспериментам и рискованным решениям, содействует извлечению уроков как из успехов организации, так и из ее неудач. Кроме того, постоянно сокращая петлю обратной связи, мы создаем все более безопасные системы и имеем возможность активнее принимать рискованные решения и проводить эксперименты, помогающие учиться быстрее конкурентов и выигрывать на рынке.

Конец ознакомительного фрагмента.