Содержание

Об авторе

Олег Скрынник – управляющий партнер компании Cleverics.

В области информационных технологий работает более 20 лет, в том числе на руководящих должностях более 15 лет. Имеет опыт построения и реорганизации работы подразделений ИТ нескольких крупных компаний (предприятия сферы услуг, финансовые учреждения, промышленные предприятия). Постоянно использует этот опыт и знания при выполнении проектов, а также с удовольствием делится ими со слушателями курсов, мастер-классов и деловых игр.

Соучредитель некоммерческого партнерства «Форум по ИТ-сервис-менеджменту» (itSMF Russia). Имеет публикации в профильных СМИ, регулярно выступает на конференциях и семинарах. Ведет свой блог на портале Real ITSM (www.realitsm.ru).

Победитель конкурса статей «ITSM в России – 2014» (1-е место).

Победитель конкурса статей «ITSM в России – 2017» (1-е место).

Формальная сертификация:

- EXIN DevOps Master;
- ITIL® Expert;
- IT Service Manager;
- Microsoft® Certified Systems Engineer;
- Accredited GamingWorks™ Trainer.

Благодарности

Я очень благодарен своей семье и друзьям. Не стану утверждать, что они сильно помогли в написании книги – к счастью, мои близкие не связаны с такими вещами, как DevOps. Однако они совершенно точно пострадали: в некоторые моменты с июля по декабрь 2017 года я вдруг становился задумчивым и не реагировал на внешние раздражители, а иногда требовал соблюдения тишины по вечерам.

Я также благодарен своим коллегам по Cleverics. Организация собственного дела совместно с лучшими известными мне умами – безусловно, одно из самых значимых решений моей жизни. Чувство единства в целях и принципах, свободы в принятии решений, ответственности за результаты, ощущение плеча коллег ровно в ту минуту, когда поддержка наиболее необходима, – все это позволило выкроить немного времени для структурирования и фиксации мыслей о DevOps и превращения их в полноценную книгу.

Наконец, ничуть не менее я благодарен своим клиентам, которые не устают удивлять интересными задачами, радовать новыми вызовами, бесконечно загружать курсами, играми и семинарами, требовать лучшего и большего, буквально не давая возможности стоять на месте.

Предисловие автора

Эта книга написана ИТ-менеджером для ИТ-менеджеров. Она показывает DevOps не как явление, связанное с новыми инструментами автоматизации, приемами программирования или технологиями. Она объясняет управленческие аспекты DevOps для тех, кто профессионально занят *управлением* информационными технологиями.

Она отличается от других книг структурностью изложения (возможно – чрезмерной) и попыткой полностью охватить такое явление, как DevOps, на базовом, фундаментальном уровне. Это не означает поверхностного изложения, достаточного для первого знакомства с новой предметной областью. Под фундаментальностью подразумевается именно построение фундамента, основы – я рассказываю об истоках DevOps, неизбежности появления, ключевых предпосылках и их отражении в практиках, про сами практики и принципы, на которых они основываются.

Несмотря на обилие литературы по данной теме, это та самая книга, которой мне очень не хватало, когда я сам изучал DevOps. Я вижу своей задачей максимально четкое, структурное и лаконичное рассмотрение непростой, но очень интересной темы. Смею надеяться, что в данной книге нет ни одного лишнего слова, и напротив – есть все необходимые слова.

Москва, 2019

Глава 1

Что такое DevOps

Методы управления ИТ-деятельностью не стоят на месте. Несколько десятков лет назад использовались одни подходы к разработке и эксплуатации информационных систем, сегодня – уже другие, а завтра придет время следующих, переосмысленных способов и техник, опирающихся на новые знания, опыт и технологии. Большую часть времени методы управления развиваются эволюционно, путем систематизации и оттачивания созданных ранее моделей, основанных на неких базовых принципах и постулатах. Однако время от времени происходят скачкообразные изменения, позволяющие отдельным организациям-лидерам сделать существенный шаг вперед в вопросах эффективности и рациональности применения информационных технологий.

В качестве наглядного примера можно привести переход в управлении ИТ-деятельностью от принципа управления ИТ-системами к управлению ИТ-услугами. Начавшись около 20 лет назад, это изменение взгляда на менеджмент дало возможность первопроходцам получить значительные конкурентные преимущества. Появляющиеся новые практики после апробации становились так называемыми лучшими практиками, используемыми лидерами. Часть лучших практик постепенно переходила в статус общепринятых норм, некоторые из которых принимали вид отраслевых стандартов (рис. 1.1). Разумеется, определенная часть организаций не использовала в своей работе ни лучшие практики, ни стандарты – пока еще не все сферы экономической деятельности испытывают существенную зависимость от информационных технологий.

Рассмотрим, к примеру, управление ИТ-услугами. В 80–90-е годы XX века возникла идея предоставления ценности от применения информационных технологий в виде услуг и организации ИТ-деятельности в форме процессов. Отдельные европейские компании стали первопроходцами, разрабатывая новые практики организации работы и подходы к решению управленческих задач. Часть таких практик, например выделение функции Service Desk, отделение инцидентов от проблем, управляемая и контролируемая обработка изменений в ИТ-инфраструктуре и др., в 2000–2001 годах была сформулирована в ключевых публикациях, таких как библиотека ITIL[1]. Это позволило им

[1] https://www.axelos.com/best-practice-solutions/itil.

перейти в разряд лучших практик и начать использоваться не только лидерами, но и «догоняющими» организациями. Со временем, в 2002 году, появился первый стандарт BS 15000–1:2002 «Управление ИТ-услугами», задавший определенную норму, которой должны следовать те, кто стремится к построению целостной системы управления ИТ-услугами. При этом практики, публикации и стандарты не останавливаются в своем развитии (рис. 1.2).

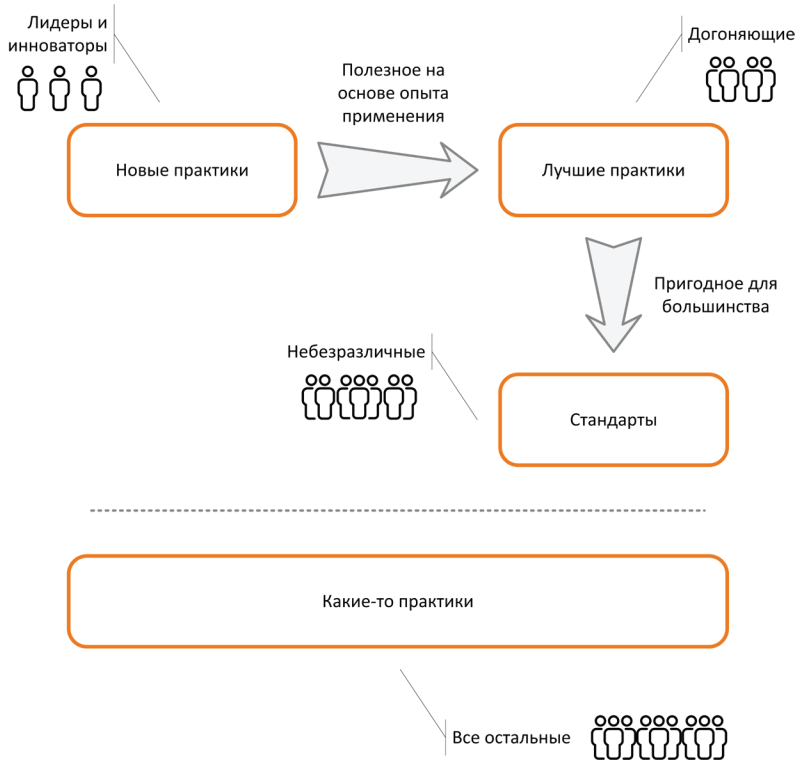

Рис. 1.1 ❖ Появление и использование новых практик

Аналогичная история происходит в настоящее время в развитии гибких подходов к разработке программного обеспечения. Однако назревающая здесь революция затрагивает более значительную область, чем только разработка ПО, и по масштабу последствий, возможно, встанет в один ряд с ITSM (IT Service Management, управление ИТ-услугами).

Новые, появляющиеся практики получили этикетку «DevOps»[1] – настолько же далекую от вкладываемого смысла, насколько ITIL далек от понятия «библиотека», а COBIT – от целей контроля.

[1] Сокр. от англ. Development & Operations.

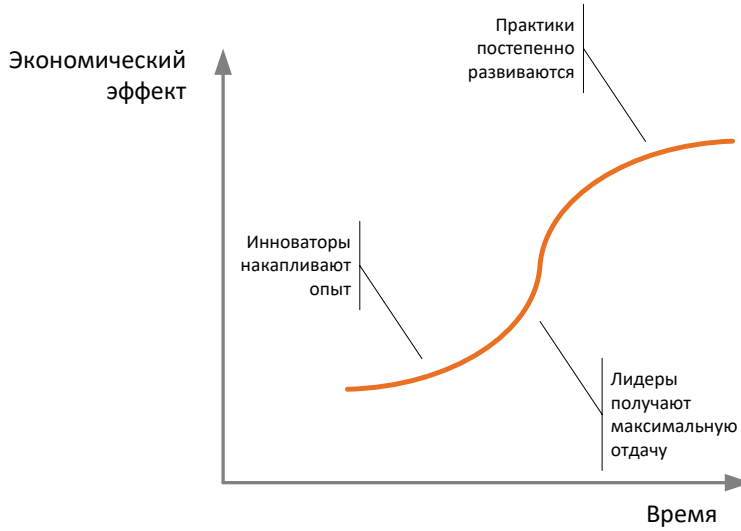

Рис. 1.2 ❖ Развитие практик

С публикацией COBIT 5 в 2012 году правообладатель подчеркнул, что, несмотря на то что изначально аббревиатура COBIT являлась сокращением от «Control Objectives for Information and related Technology», теперь она является именем собственным[1].

Компания AXELOS, управляющая ITIL с 2013 года, также не рекомендует использовать первоначальное наименование «IT Infrastructure Library», ограничиваясь именем собственным ITIL.

Эксперты DevOps, стоявшие у истоков этого движения, признают ограниченность получившегося названия, призывая использовать более точные, на их взгляд, «BizDevOps», «DevSecOps» и подобные. Однако вероятность изменения названия в настоящее время является незначительной.

Тем не менее само явление весьма достойно изучения. Для полного понимания сути DevOps необходимо рассмотреть предпосылки к появлению как идеи, так и связанного с ней движения.

Истоки

Можно утверждать, что появлению DevOps в наибольшей степени способствовали два фактора: развитие гибких методов разработки программного обеспечения и управление ИТ-инфраструктурой как программным кодом. Рассмотрим каждый из них.

[1] http://www.isaca.org/COBIT/Pages/FAQs.aspx.

Развитие гибких методов разработки программного обеспечения

В конце прошлого столетия доминирующей методологией разработки программного обеспечения была так называемая «водопадная модель» (рис. 1.3): последовательное выполнение заранее определенных этапов, каждый из которых занимает существенное время и завершается достижением заранее согласованных результатов, при этом переход на следующий этап во многих случаях происходит после полного и формального завершения предыдущего этапа. Дополнительным отличительным признаком такой модели является функциональная специализация исполнителей отдельных этапов: аналитиков, архитекторов, разработчиков, тестировщиков и т. д.

Рис. 1.3 ❖ Пример водопадной модели разработки программного обеспечения

При разработке крупных информационных систем с конечной функциональностью, которую можно определить и зафиксировать в самом начале работ, а также при отсутствии требования максимально быстрого завершения полного цикла разработки такая модель позволяет получать качественные выходные результаты при достаточно детальном контроле расходов.

Однако в конце 90-х годов XX века, с бурным развитием интернет-технологий и веб-программирования, недостатки водопадной модели стали негативно влиять на взаимодействие и взаимопонимание заказчиков (бизнес-подразделений компании, либо внешних организаций) и исполнителей (программистов компании, либо внешних разработчиков программного обеспечения). Действительно, появляющиеся рыночные возможности для основного бизнеса требовали быстрого, за считанные месяцы, вывода на рынок новых продуктов. В то время как типичный цикл разработки от начала проекта до получения первого работающего результата занимал от 6 до 18 месяцев, в крупных организациях – до 2–3 лет. Кроме того, в условиях появления ранее неизвестных, но потенциально перспективных рыночных возможностей требования заказчиков могли меняться по ходу проекта разработки, что было крайне сложно учесть при создании ИТ-системы без увеличения сроков либо снижения качества выходных результатов (рис. 1.4).

Рис. 1.4 ❖ Классическая пирамида взаимосвязанных ограничений проектного управления

Таким образом, накапливалось напряжение между заказчиками и исполнителями, между основным бизнесом и разработчиками ПО. Ответом на такой вызов стали инновационные подходы к программированию. Кен Швейбер (Ken Schwaber) выпустил несколько публикаций о Scrum[1]. Кент Бек (Kent Beck) опубликовал книгу об экстремальном программировании, XP[2]. Однако применение новых идей давало весьма скромные результаты, в основном потому, что такое применение фокусировалось лишь на одном из этапов цикла разработки ПО – на собственно программировании, притом что задача ставилась намного более широкая. Требовалось что-то, что позволит упростить и ускорить весь жизненный цикл программного обеспечения.

---

[1] Например, *Schwaber K.* Agile Software Development with Scrum. 2001. ISBN: 978-0130676344.

[2] *Beck K.* Extreme Programming Explained: Embrace Change. 1999. ASIN: B01FKT01PY.

В 2001 году К. Швейбер, К. Бек, а также еще пятнадцать экспертов встретились, чтобы обсудить имевшиеся проблемы и выработать решение. Итогом стал так называемый манифест Agile, призванный устранить разрыв понимания между бизнесом и разработчиками ПО. Один из авторов манифеста, Роберт Мартин (Robert C. Martin), поясняет[1]:

> *«При использовании правильных дисциплин и правильного минимального процесса может возникнуть и развиться доверие между разработчиками и бизнесом. Бизнес начнет доверять разработчикам, вместо того чтобы думать, что они ленивые, продажные, противные существа, а разработчики начнут обращать внимание на бизнес и осознают, что те являются разумными, рациональными существами, а не с какой-то другой планеты».*

Последовавшее развитие и принятие идей гибкой разработки сообществом программистов и менеджеров проектов сильно ускорили и перестроили разработку ПО.

Манифест Agile
(полный текст, цитируется по официальному переводу[2])

Люди и взаимодействие	важнее	процессов и инструментов
Работающий продукт	важнее	исчерпывающей документации
Сотрудничество с заказчиком	важнее	согласования условий контракта
Готовность к изменениям	важнее	следования первоначальному плану

То есть, не отрицая важности того, что справа, мы все-таки больше ценим то, что слева. Мы следуем таким принципам.

1. Наивысшим приоритетом для нас является удовлетворение потребностей заказчика благодаря регулярной и ранней поставке ценного программного обеспечения.
2. Изменение требований приветствуется, даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.
3. Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.
4. На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.
5. Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.
6. Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.
7. Работающий продукт – основной показатель прогресса.

---

[1] https://www.youtube.com/watch?v=hG4LH6P8Syk, а также https://www.aaron-gray.com/a-criticism-of-scrum/.

[2] http://agilemanifesto.org/iso/ru/manifesto.html.

8. Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки.
9. Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.
10. Простота – искусство минимизации лишней работы – крайне необходима.
11. Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.
12. Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Ключевыми элементами гибкой разработки являются более плотное взаимодействие между заказчиком и исполнителем, уменьшение размера задач, ритмичность выдачи результатов через короткие промежутки времени (циклы) и ограничение размера команд.

Группа разработки ПО, применяющая гибкие подходы, выдает готовый к эксплуатации новый код каждые две–четыре недели. Конечные потребители плотнее вовлечены в создание продукта, а значит, быстрая обратная связь значительно влияет на дальнейшее развитие продукта, что дополнительно добавляет вкуса к быстрым изменениям.

Однако во многих компаниях отказ от водопадной модели в пользу гибкой разработки дает куда меньший эффект, чем ожидается. Такие наблюдаемые в работе многих компаний результаты связаны не столько с какими-то преимуществами водопадной модели или недостатками Agile. Зачастую полезный эффект нивелируется тем, что разработка кода – лишь одно из звеньев в цепочке создания ценности.

Действительно, до начала собственно разработки имеется еще значительный блок работ, направленный на выявление бизнес-потребностей, их проработку, анализ, приоритизацию и т. д.

Далее, по окончании разработки готовый программный код необходимо быстро развернуть в среде эксплуатации, чтобы заказчики получили всю ту пользу, которую им обещали, а также могли предоставить обратную связь разработчикам относительно качества получившегося продукта. При этом почти во всех организациях, возникших до 2010-х годов, ИТ-инфраструктура является жесткой, основанной на дорогом аппаратном обеспечении, которое было приобретено достаточно давно, бюджеты на закупку и настройку выделялись непросто, да и собственно бюджетный процесс для новых закупок долгий.

Более того, в подавляющем числе организаций ИТ-инфраструктура находится в довольно хрупком состоянии. Одним из факторов, усиливающих такую хрупкость, является комплексность, сложность применяемых ИТ-решений. Используется множество, десятки тысяч взаимосвязанных компонентов. Другим фактором служит слабое документирование, равно как и быстрое устаревание документации относительно применяемых ИТ-решений и ИТ-систем, в том числе устаревание знаний ИТ-персонала, а также потеря знаний вследствие текучки кадров.

Трогать ИТ-инфраструктуру во многих компаниях страшно. Изменение – самое большое зло для отдела эксплуатации ИТ-систем, а постоянный большой поток изменений может привести к катастрофическим последствиям.

Таким образом, передовые методы разработки ПО упираются в барьеры со стороны подразделений, ответственных за эксплуатацию информационных технологий, что нивелирует возможный положительный эффект применения гибких подходов.

> Для борьбы с хрупкостью некоторые организации применяют формализованный и автоматизированный процесс управления изменениями, призванный структурировать поток изменений и минимизировать риски при их выполнении.

Управление ИТ-инфраструктурой как программным кодом

Возникновению управления ИТ-инфраструктурой как программным кодом предшествовало появление и развитие двух технологий: виртуализации и облачных вычислений.

История виртуализации программных и аппаратных сред началась довольно давно, в 1964 году, с началом разработки операционной системы IBM CP-40[1]. За годы последовательного развития этого направления был достигнут весьма значительный прогресс. Коммерчески доступные системы появились для мейнфреймов (70–80-е годы прошлого века) и для более распространенных в последующем машин на архитектуре Intel x86 (80–90-е годы)[2]. Рисунок 1.5 показывает количество ключевых событий, связанных с виртуализацией, на отрезке с 1964 по 2008 год (график не случайно завершается данным годом, что станет видно из дальнейшего изложения).

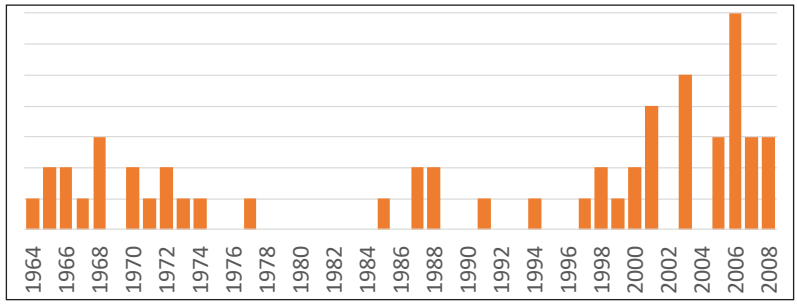

Рис. 1.5 ❖ Распределение во времени ключевых событий, связанных с виртуализацией

[1] https://en.wikipedia.org/wiki/IBM_CP-40.

[2] Интересно отметить, что по утверждению Д. Хамбла в те годы случился некоторый период времени, когда компания IBM избегала рекомендовать продукты виртуализации своим клиентам, так как это влияло на продажи оборудования.

Виртуализация позволила не только более эффективно использовать дорогое и мощное аппаратное обеспечение, но и ввести дополнительный уровень абстракции между исполняемым кодом, предоставляющим полезные результаты заказчику, и нижележащим системным программным обеспечением. Был сделан существенный шаг в сторону разделения компетенции и ответственности между, условно говоря, «прикладниками» и «системщиками», в широком смысле данных понятий.

Технология облачных вычислений развивалась еще быстрее. До середины 90-х годов прошлого века телекоммуникационные компании предлагали своим клиентам организацию частных глобальных вычислительных сетей (WAN, Wide Area Network) путем прокладывания соответствующих соединительных кабелей для каждой точки, каждого заказчика, от пункта А до пункта Б. Но с появлением технологии частных виртуальных сетей (VPN, Virtual Private Network) возникла возможность по одним и тем же каналам передачи данных отправлять пакеты разных клиентов, обеспечивая должный уровень безопасности, приватности и качества сервиса. Именно тогда для наглядного отображения разграничения ответственности – где идет «кабель от клиента», а где трафик попадает в общую разделяемую сеть, провайдеры стали использовать символ облака.

Клиенты, получившие возможность передачи данных на большие расстояния, стали использовать данные технологии не только для собственно обмена информацией между своими территориально удаленными друг от друга системами, но и для распределения вычислительной нагрузки между разными узлами своих сетей. Напрашивалось появление технологии, упрощающей и удешевляющей такое взаимодействие. Небольшие провайдеры сделали первые шаги, а действительно масштабные изменения случились в 2006 году, когда компания Amazon представила свое решение ECC (Elastic Compute Cloud). Вскоре, в 2008 году, компания Microsoft запустила свой сервис, Azure, а компания Google – сервис Google App Engine, впоследствии развившийся в Google Cloud Platform. Это, разумеется, не единственные, но самые крупные примеры предоставления вычислительных мощностей в аренду.

Виртуализация и облачные технологии сильно изменили вычислительный ландшафт. Предлагаемые коммерческими провайдерами ресурсы стали доступными по стоимости, надежными и обеспечивающими необходимый уровень безопасности. Отношение клиентов к облакам и их использованию изменилось от «*кто-то другой где-то управляет моим железом*» на «*у меня есть инфраструктура, которой я управляю на расстоянии*».

Национальный институт стандартов и технологий США определил пять ключевых характеристик облачных вычислений[1]:

1) самообслуживание по требованию – потребитель самостоятельно определяет и изменяет вычислительные потребности, такие как серверное время, скоро-

[1] http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

сти доступа и обработки данных, объем хранимых данных без взаимодействия с представителем поставщика услуг;
2) универсальный доступ через сеть – услуги доступны потребителям по сети передачи данных через стандартные механизмы, которые способствуют использованию гетерогенными тонкими или полными клиентскими платформами;
3) объединение ресурсов – поставщик услуг объединяет ресурсы для обслуживания большого числа потребителей в единый пул для динамического перераспределения мощностей между потребителями в условиях постоянного изменения спроса на мощности;
4) эластичность – услуги могут быть предоставлены, расширены, сужены в любой момент времени, без дополнительных издержек на взаимодействие с поставщиком, как правило, в автоматическом режиме, соизмеримо со спросом;
5) учет потребления – поставщик услуг автоматически измеряет потребленные ресурсы на определенном уровне абстракции и на основе этих данных оценивает объем предоставленных потребителям услуг.

Что же это означает – управлять инфраструктурой на расстоянии? Вспомним одну из ключевых парадигм Unix-систем: все необходимые действия с системой можно произвести из командной строки (а значит, и с помощью скрипта). Графические оболочки являются красивым, но опциональным инструментом.

Объединим теперь виртуальные облачные технологии и интерфейс командной строки для всех задач. В результате ИТ-специалисты получили возможность с помощью текстовых команд создавать необходимые части ИТ-инфраструктуры, включая серверы, системы хранения данных, сетевые компоненты, все интерфейсы между ними, все настройки и конфигурации… Степень автоматизации существенно возросла, равно как и скорость выполнения необходимых изменений. Раньше для разворачивания ИТ-инфраструктуры, основанной на собственном аппаратном обеспечении, требовалось:

- обосновать и согласовать бюджет (недели и месяцы);
- дождаться очередного цикла закупки (месяцы);
- заказать оборудование у поставщика и оплатить его (дни);
- дождаться поставки (недели и месяцы);
- получить, установить, настроить, подготовить к использованию (дни и недели).

Теперь аналогичную по характеристикам ИТ-инфраструктуру можно создать так:

- запустить скрипт, дождаться окончания его выполнения (минуты, редко – часы);
- оплатить счет облачного провайдера в конце месяца.

То есть необходимая инфраструктура создается с помощью программного кода. И не только создается, но и может управляться как программный код – с хранением версий, отслеживанием изменений, отладкой, повторным использованием прошлых наработок и т. д. Более подробно данные аспекты будут рассмотрены в главе 3 «Принципы».

В завершение отметим также вторую жизнь, которую получили давно придуманные технологии. К примеру, виртуализация на уровне операционной системы была доступна во многих UNIX-системах еще в 80-е годы прошлого столетия. Однако серьезный коммерческий успех этой технологии, которую чаще стали называть контейнеризацией, пришел только во второй половине 2000-х, что совпадает по времени с событиями, описанными ранее. И если изначальный механизм chroot был довольно ограничен по функциональности и возможностям, то сейчас для контейнеров можно изолировать файловую систему, выделять дисковые квоты, ограничивать предоставляемые оперативную память, время процессора, ширину каналов ввода-вывода и т. д.

Неизбежность появления

> «Когда вы слышите, что кто-то говорит о неизбежности, – скорее всего, за этим скрываются коммерческие компании, очень старающиеся сказку сделать былью».
>
> *Ричард Столман (Richard Stallman), основатель Free Software Foundation и создатель операционной системы GNU, об облачных вычислениях*[1]*, 2008*

Рассмотренные истоки возникновения DevOps позволяют сделать следующие выводы.

Во-первых, из-за появления новых способов взаимодействия с основным бизнесом, клиентами и грамотного применения методов гибкой разработки назрела *потребность* строить работу и управление информационными технологиями иначе.

Во-вторых, с возникновением новых технологий управления инфраструктурой появилась *возможность* строить работу ИТ иначе.

Трезво воспринимая слова Р. Столмана, приведенные выше (а с облачными вычислениями он, похоже, сильно ошибся), можно предполагать, что появление чего-то, аналогичного DevOps, было лишь вопросом времени.

Определение

Только очень самоуверенные либо бесконечно некомпетентные люди, а также общепризнанные гуру могут всерьез рассуждать о каком-либо явлении, не дав ему предварительно определения либо не опираясь на общепринятое определение. С DevOps, к сожалению, ситуация отнюдь не проста.

Некоторые эксперты пытаются придумать что-то свое, близкое их пониманию. Другие утверждают, что определить DevOps в настоящий момент невозможно, так как это, скорее, явление, движение, идея, но не дисциплина и не

[1] https://www.theguardian.com/technology/2008/sep/29/cloud.computing.richard.stallman.

методология. Третьи сообщают, что DevOps у каждого свой, и приводят известную аналогию про нескольких слепых, ощупывающих слона: один говорит, что, скорее всего, это дерево, другой – что коврик, третий – что змея, и т. д.

За то время, которое я занимаюсь данной темой, мне удалось прочесть большое количество книг и интернет-публикаций, пообщаться с совершенно разными людьми, вовлеченными в DevOps-движение – как в России, так и в Европе, посетить тематические учебные курсы и сдать несколько международных профильных экзаменов. На мой взгляд, опасения по поводу невозможности определения DevOps несколько преувеличены. Разумеется, сколько людей – столько и мнений, а в случае с консультантами все еще серьезнее: два консультанта – минимум три мнения. Однако наличие системного склада ума, высшего технического образования и опыта консультирования в сфере ИТ-менеджмента дает возможность подойти к вопросу четко и структурно. Не претендуя на универсальность или истину в последней инстанции, я составил следующее определение:

> *DevOps – продолжение идей гибкой разработки программного обеспечения и бережливого производства, примененное к полной цепочке создания ценности в ИТ, позволяющее добиваться большего от современных информационных технологий за счет культурных, организационных и инструментальных изменений.*

В данном определении есть важные моменты, которые необходимо подчеркнуть.

Во-первых, представляется важным указать на то, что DevOps не подменяет собой гибкие и бережливые практики, но как бы вбирает их в себя. Общение с коллегами, клиентами, слушателями учебных курсов показывает, что те, кто незнаком с идеями гибкой разработки, открывают для себя в DevOps очень много нового и любопытного. Те же, кто имеет соответствующую подготовку и опыт, удивляются большому количеству пересечений между практиками DevOps и практиками, скажем, Lean, Scrum и Kanban. На мой взгляд, не совсем корректно называть такое явление «пересечением». Скорее, речь про заимствование, расширение идей гибкой разработки и бережливого производства. Более подробно данный вопрос рассматривается в главе 2 «Фундамент».

Во-вторых, сама суть DevOps заключается в том, что ИТ-подразделение вместе с бизнесом думает не только про разработку программного обеспечения, но про всю цепочку создания ценности. Эта цепочка начинается с генерации новых идей совместно с представителями бизнес-подразделений, проходит через разработку, тестирование, развертывание, вплоть до эксплуатации. Такой взгляд дает возможность системного анализа, поиска и устранения узких мест во всей цепочке, налаживания механизмов предоставления обратной связи не только из конца цепочки в ее начало, но и внутри, между шагами, равно как и в пределах одного шага. Перечисленным следствиям из данного определения – системному взгляду, работе с ограничениями и организации обратной

связи – в DevOps уделяется максимальное внимание, о чем будет подробно рассказано позже.

В-третьих, следует сделать акцент на ожидаемой пользе от применения DevOps, которая заключается в большей отдаче от информационных технологий. Согласно классическому представлению, информационные технологии позволяют организациям получать больше выгод (через создание новых возможностей или устранение имеющихся ограничений), снижать риски и оптимизировать ресурсы. Правильно построенный DevOps позволяет учесть все три перечисленных аспекта. Некорректно было бы утверждать, что организации не могут получать пользу от информационных технологий традиционными способами, без применения DevOps. Однако DevOps позволяет получить *больше* пользы, которая может выражаться в ускорении вывода на рынок новых и модифицированных продуктов, быстрой реакции на потребности клиентов, увеличении доступности и устойчивости ИТ-систем, более эффективном использовании ограниченных ресурсов. Несколько подробнее данная тема будет изложена в разделе «Зачем нужен DevOps».

И наконец, в завершающей части определения есть явное указание на три ключевые составляющие: культурные, организационные и инструментальные средства (рис. 1.6). По сути, это та самая старая мантра про процессы, людей и технологии. Опыт первопроходцев DevOps, а также их последователей показывает, что ее важность все так же сильна.

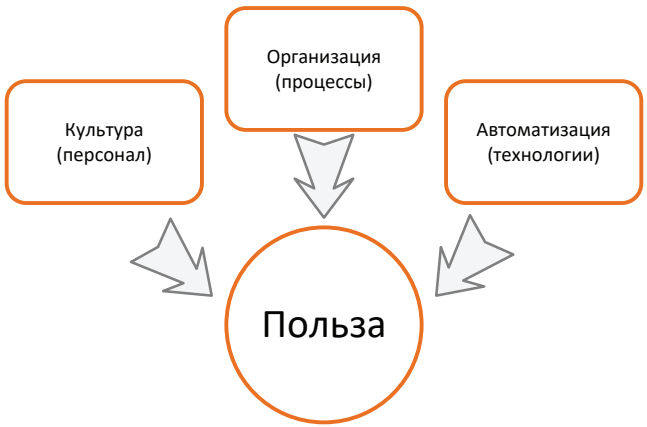

Рис. 1.6 ❖ Три ключевых компонента

Приведем данное выше определение с разбивкой на существенные составные части, позволяющей еще раз сделать необходимые акценты.

DevOps:

а) *продолжение идей гибкой разработки программного обеспечения и бережливого производства*;

б) *примененное к полной цепочке создания ценности в ИТ*;
в) *позволяющее добиваться большего от современных информационных технологий*;
г) *за счет культурных, организационных и инструментальных изменений.*

Зачем нужен DevOps

> «Послушайте! Ведь если звезды зажигают – значит, это кому-нибудь нужно?»
>
> *В. Маяковский*[1]*, 1914*

Некоторые подходы к управлению появляются как отражение фантазии автора (возможно, эксперта, или даже гуру) – этакие теоретические изыскания. Их применимость либо неприменимость доказывается адептами и последователями, пытающимися использовать новые приемы в своей работе и в организации работы других сотрудников.

Иные подходы рождаются как ответ на вполне насущные потребности. Они создаются не по заказу Британской короны и не группами специально привлеченных консультантов, но практиками, ищущими возможности устранить какие-либо сложности или ограничения, либо более эффективно использовать имеющиеся ограниченные ресурсы, либо создать новые бизнесы, новые ниши, новые инструменты решения управленческих задач.

Похоже, что DevOps ближе ко второй группе, нежели к первой. Подробности появления будут рассмотрены в разделе «История возникновения»; пока же сконцентрируемся на основных проблемах, которые различные организации пытаются решить с помощью DevOps.

Уменьшение времени вывода на рынок

Компании, применяющие DevOps, наиболее часто сообщают о необходимости существенно сокращать время вывода на рынок (англ. Time to market). Под этим термином разные люди подразумевают разное. Часто встречающееся понимание – время от зарождения какой-либо бизнес-идеи до возможности клиенту приобрести новый продукт или получить новую услугу, являющуюся результатом воплощения бизнес-идеи в жизнь. Таким образом, в расчет (а точнее – в оценку) времени вывода на рынок включается довольно большой промежуток, содержащий в случае необходимости привлечения ИТ-департамента следующие шаги:

- структурирование и первое формальное описание бизнес-идеи, а скорее – нескольких бизнес-идей, их обоснование;
- оценка и выбор бизнес-идеи для реализации;

[1] https://ru.wikisource.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D1%83%D1%88%D0%B0%D0%B9%D1%82%D0%B5!_(%D0%9C%D0%B0%D1%8F%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%B8%D0%B9).

- планирование необходимых действий для реализации, выделение финансирования;
- подготовка бизнес-процессов и персонала;
- одновременно с этим: формализация требований, разработка прототипа, первичное тестирование, разработка полнофункциональной ИТ-системы, ее тщательное тестирование, передача в эксплуатацию, запуск, тиражирование;
- одновременно с этим: маркетинговые активности, подготовка рынка, подготовка механизма и каналов продаж;
- запуск нового продукта или новой услуги.

Описанному процессу присущи некоторые сложности. Во-первых, его общая длительность может составлять годы, притом что бизнесу хотелось бы сократить ее до месяцев. Бизнес-обоснование здесь прозрачно: за время разработки нового продукта рынок может измениться настолько, что собственно продукт будет уже неактуален либо конкуренты выпустят аналогичный продукт раньше, соберут сливки и закрепятся как лидеры. Ранний выход на рынок с привлекательным конкурентным предложением помогает занять доминирующее положение в новых нишах, которое, в свою очередь, дает лидеру возможности в дальнейшем изменять рынок, подстраивая его под себя. Это – существенное преимущество, которым обладают немногие, хотя стремятся к нему все. Кроме того, не следует забывать про все возрастающую скорость изменений. Одна из наиболее наглядных иллюстраций данного тезиса – закон ускоряющихся возвратов (Law of Accelerating Returns), сформулированный в 1999 году Реем Курцвайлом[1] (Ray Kurzweil). Согласно ему, скорость изменений в широком спектре эволюционирующих систем, включая новые технологии, но не ограничиваясь ими, стремится расти экспоненциально. На практике это означает, что прорывы в технологиях, в том числе информационных, случаются все чаще. Компании, которые *увеличивают* темп изменений, становятся лидерами, а те, кто лишь *могут сохранять* свой быстрый темп, получают возможность не остаться на обочине. Что уж говорить про тех, кто не способен меняться быстро…

> «Авторам сценариев для кинематографа приходится туго. За меняющейся с быстротой молнии обстановкой положительно не угнаться. …Пока успеешь написать сценарий, пока его разыграют, пустят по прокатным конторам и пока он дойдет до экрана, смотришь: выпархивает что-либо более новое, более отвечающее моменту…
>
> В точно таком же положении оказываются авторы… разных и злободневных пьесок. Им придется скоро творить прямо за утренним кофе, за чтением газетных телеграмм. С тем чтобы к полудню пьеса была готова для генеральной репетиции, а вечером появлялась перед публикой. Только при этом непременном условии можно даже уловить момент».
>
> «Театр», московская ежедневная театральная газета[2], август 1917

[1] http://www.kurzweilai.net/the-law-of-accelerating-returns.

[2] https://project1917.ru/groups/teatr.

Вторая сложность описанного выше процесса заключается в необходимости четкой координации и согласования взаимозависимых шагов, особенно выполняющихся параллельно. В этот момент многие компании попадают в классическую ловушку: пока нет готового продукта – нечего рекламировать и продавать, однако с появлением такового начало маркетинговых активностей приводит к продажам (а значит – и к финансовой отдаче) лишь с задержкой. Такая ловушка еще больше увеличивает фактическое время вывода на рынок и требует еще более аккуратной координации всех исполнителей.

Отметим, что роль традиционного ИТ-подразделения в увеличении срока вывода на рынок трудно переоценить. Действительно, в некоторых организациях из общего календарного времени в 1,5–2 года на ИТ-работы приходится более 50–70%.

Другое понимание термина «время вывода на рынок» менее глобально, но не менее значимо. Динамичные компании, создающие цифровые продукты, привыкли действовать быстро. Скрупулезному и детальному планированию они предпочитают эксперименты, а слово «идея» заменяют на «гипотезу». В этом случае процесс выглядит примерно так:

- рождение гипотезы, разработка методов оценки ее справедливости;
- реализация гипотезы на практике;
- измерение результата, A/B-тестирование, сравнение с целевыми значениями;
- корректировка по итогам анализа, переход на первый или второй шаг.

Несложно заметить возникновение цикла, ожидаемая скорость которого – недели. Такой быстрый темп необходим потому, что сама суть движения – в постоянном поиске. На старте, в самом начале, совершенно неизвестно конечное состояние, и тем более неизвестна дорога к нему. Долгосрочное планирование не имеет никакого смысла, компания видит лишь следующий, ближайший шаг – точнее, пытается его угадать. Проиллюстрировать данный тезис поможет широко известная метафора, сравнивающая выживание и развитие бизнеса с поиском реки с деньгами (рис. 1.7). Один раз войдя в такую реку, нащупав новую нишу и новые возможности, компании будет необходимо всегда искать изменяющееся русло. Притом, что традиционные процессы, регламенты, уже имеющиеся продукты будут с большой вероятностью увеличивать инерцию компании и, оставленные без внимания, приведут к выходу на берег.

Нетрудно догадаться, что вклад ИТ-департамента в замедление приведенного выше цикла высок. Действительно, в создании цифровых продуктов роль ИТ – ключевая, поэтому задержки на этапе реализации гипотезы в наибольшей степени происходят именно благодаря «медленному» ИТ-отделу, предлагающему вместо ожидаемых недель – месяцы.

Для уменьшения времени вывода на рынок DevOps предлагает множество техник, например: уменьшение размера задач, уменьшение количества передач работы, постоянные поиск и устранение потерь и др. Они будут более подробно рассмотрены в главе 4 «Основные практики». Сейчас же важно сделать следующее замечание: наивно надеяться, что применение техник DevOps для

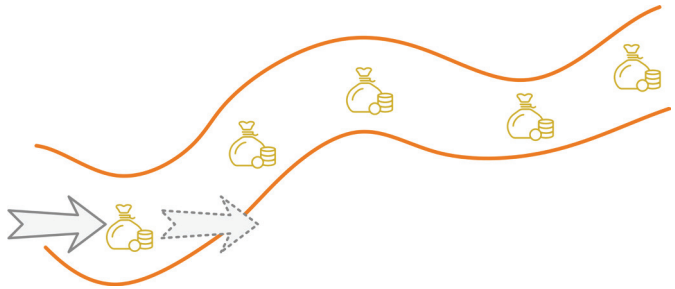

Рис. 1.7 ❖ Река с деньгами

ускорения работы ИТ-отдела одновременно приведет к сокращению затрат на ИТ. Скорее наоборот, расходы на информационные технологии вырастут, что обусловлено в первую очередь увеличением количества ИТ-персонала. Действительно, традиционная организация ИТ-отдела предполагает наличие отдельных функциональных подразделений, каждое из которых занимается всеми задачами в рамках своей предметной области (бизнес-анализ, разработка и тестирование, эксплуатация, поддержка, развитие и т. д.). При этом внутри каждого такого функционального подразделения обеспечивается необходимая взаимозаменяемость специалистов, а среднее и большое число специалистов одинаковых квалификации и компетенций позволяет равномерно распределять между ними нагрузку (рис. 1.8).

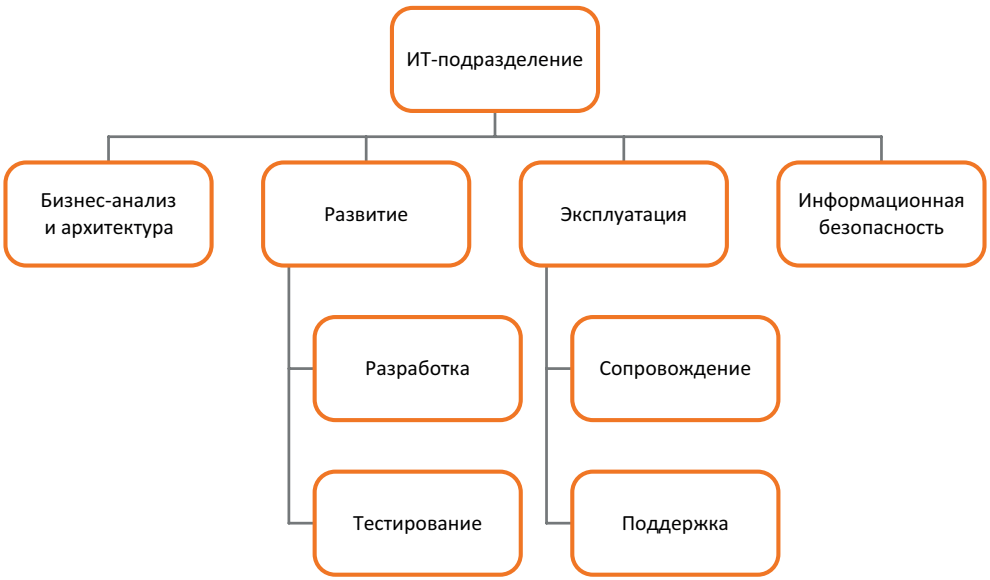

Рис. 1.8 ❖ Функциональная структура традиционного ИТ-подразделения