

ACCELERATE

# State of DevOps 2019

РУССКАЯ ВЕРСИЯ

Sponsored by



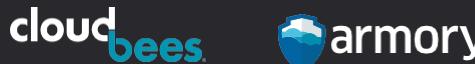
Google Cloud

Pivotal  
cloud  
**bees**

Deloitte.

sumo logic

redgate



# СОДЕРЖАНИЕ

КРАТКИЙ ОБЗОР	3	ЗАКЛЮЧЕНИЕ	76
ОСНОВНЫЕ ВЫВОДЫ	5		
КТО УЧАСТВОВАЛ В ИССЛЕДОВАНИИ?	7	МЕТОДОЛОГИЯ	77
ДЕМОГРАФИЯ И СТАТИСТИКА ПО ОРГАНИЗАЦИЯМ	8		
КАК ПРОВОДИЛОСЬ СРАВНЕНИЕ?	14	БЛАГОДАРНОСТИ	79
КАК ПОЛЬЗОВАТЬСЯ ИССЛЕДОВАТЕЛЬСКИМИ МОДЕЛЯМИ	26	ОБ АВТОРАХ	80
КАК ПОВЫСИТЬ?	29	ПРИЛОЖЕНИЕ А ВИЗУАЛЬНОЕ ПРЕДСТАВЛЕНИЕ 4 ОСНОВНЫХ МЕТРИК	81
ЭФФЕКТИВНОСТЬ ПОСТАВКИ ПО И ОПЕРАЦИОННОЙ ДЕЯТЕЛЬНОСТИ	30		
ПРОИЗВОДИТЕЛЬНОСТЬ	55		
МЕТОДИКИ ТРАНСФОРМАЦИИ: ЧТО РЕАЛЬНО РАБОТАЕТ	69	ПРИЛОЖЕНИЕ Б СТРАТЕГИЯ МАСШТАБИРОВАНИЯ	82



## КРАТКИЙ ОБЗОР

Отчет Accelerate State of DevOps составлен по итогам шести лет исследований и анализа данных, полученных от 31 000 экспертов со всего мира. Это самое крупное и продолжительное исследование такого рода, которое предлагает независимую оценку практик и подходов, обеспечивающих высокую эффективность и производительность команд разработки программного обеспечения.

Результаты исследования позволяют понять, какие практики и принципы помогают командам реализовывать свое превосходство в области разработки и поставки технологий и достигать значительных бизнес-результатов.

Опираясь на строгие методы анализа данных, мы представляем анализ самых эффективных и результативных способов разработки и поставки

технологий. Кластерный анализ позволяет командам сравнить свои показатели эффективности с отраслевыми и отнести себя к одной из категорий: «низкопроизводительные» (low performers), «среднепроизводительные» (medium performers), «высокопроизводительные» (high performers), «сверхпроизводительные» (elite performers).

Затем на основе выводов нашего прогнозного анализа они смогут выявить конкретные принципы, практики и подходы, которые позволят повысить эффективность процесса поставки программного обеспечения и в конечном счете достичь уровня выдающихся команд.

В этом году мы также исследовали способы поддержки инженерной производительности, в числе которых: возможность поиска и получения



дополнительной информации, более эффективные и удобные инструменты развертывания, сокращение технического долга благодаря гибкой архитектуре приложений, поддерживаемости кода и системам визуализации и мониторинга.

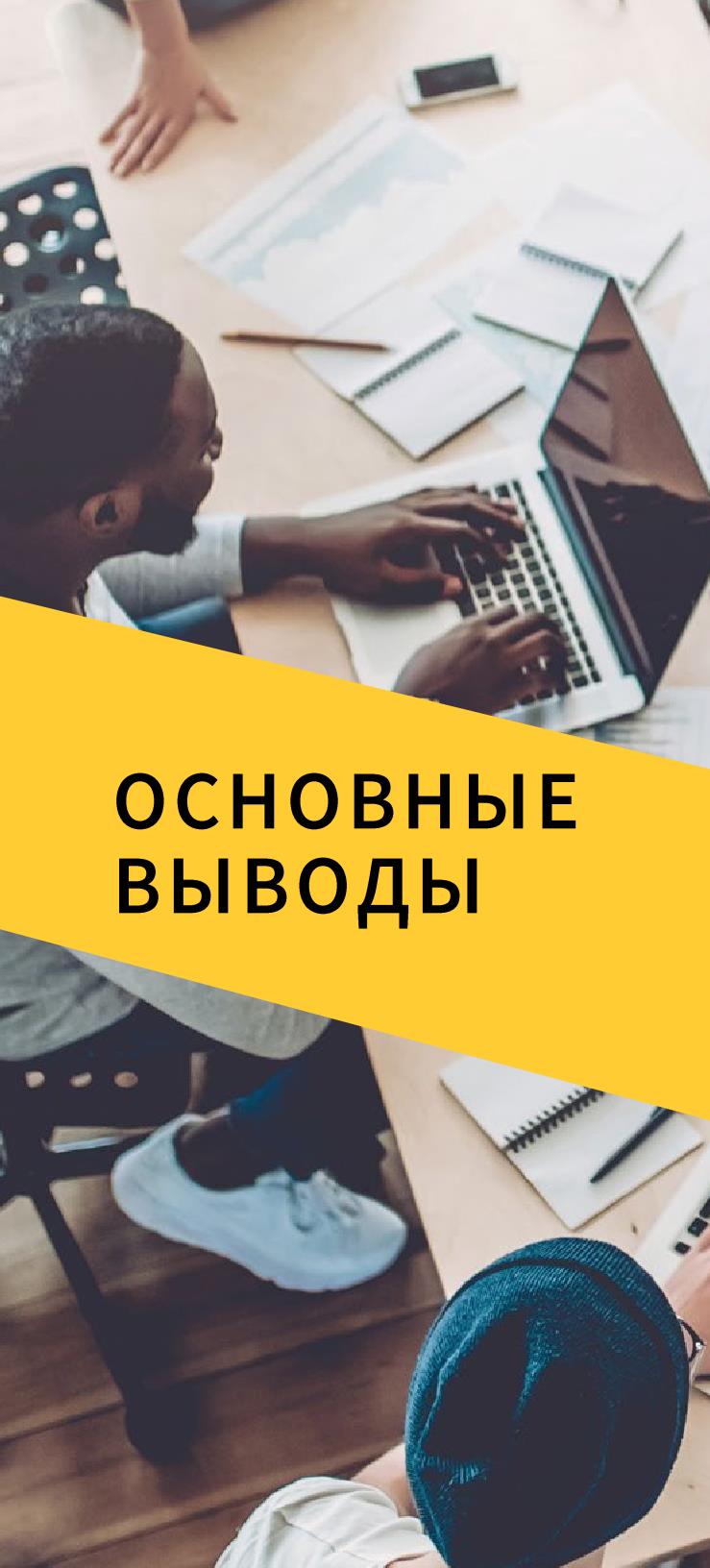
Наше исследование вновь показало, что индустриальный стандарт Четырех Основных Показателей<sup>1</sup> разработки и поставки программного обеспечения стимулирует производительность организации в технологической трансформации. Отчет этого года подтвердил предыдущие выводы о том, что высокой стабильности можно достичь без ущерба для скорости.

Также мы выяснили, какие инструменты и подходы

стимулируют улучшение четырех основных показателей. В их числе: инженерные практики, использование облачных технологий, организационные принципы (включая процедуры согласования изменений) и культурные аспекты. Для компаний, которым нужны рекомендации по улучшению процессов, мы обозначили единственный реалистичный путь: начать с основ, затем реализовать культуру непрерывного улучшения, определив одно или несколько ограничений, уникальных для организации. Когда влияние соответствующего ограничения устранено, и он больше не сдерживает развитие компании, повторите процесс. Также в отчете обозначены самые эффективные стратегии для реализации подобных изменений.

---

<sup>1</sup> <https://www.thoughtworks.com/radar/techniques/four-key-metrics>



## ОСНОВНЫЕ ВЫВОДЫ

1

**Отраслевые показатели продолжают улучшаться, особенно среди «elite performers».**

Доля самых эффективных выросла почти в три раза и теперь составляет 20% от всех команд. Это доказывает, что превосходство достижимо — те команды, кто инвестирует в ключевые принципы, практики и подходы, получают преимущества.

2

**Быстрая, надежная и безопасная доставка изменений программного обеспечения лежит в основе технологической трансформации и эффективности организаций.**

Мы по-прежнему получаем подтверждения того, что скорость доставки, стабильность и доступность программного обеспечения влияют на эффективность организаций (в том числе на рентабельность, производительность и удовлетворенность клиентов). У самых эффективных команд в два раза больше шансов достичь или даже перевыполнить цели их организаций.

**Лучшие стратегии для масштабирования DevOps в организациях строятся на структурных изменениях, формирующих внутренние сообщества.**

«High performers» предпочитают стратегии, формирующие структуры сообществ на нижнем и верхнем уровнях организации, например сообщества по практикам (Communities of Practice) и сообщества по проверки концепций (Proofs of Concept). Такие стратегии делают компании более устойчивыми к реорганизациям и изменениям в продуктах.

**Использование облачных технологий является характерной чертой выдающихся команд и стимулирует их высокую производительность.**

Использование облачных технологий, согласно статье NIST SP 800-145, является фактором эффективности поставки и доступности программного обеспечения. Самые эффективные команды по сравнению с отстающими в 24 раза чаще реализуют все пять основных характеристик в сфере облачных вычислений.

**Хорошая производительность помогает улучшить баланс между работой и личной жизнью сотрудников, а также сократить количество случаев профессионального выгорания. Организации могут делать разумные инвестиции в этом направлении.**

Чтобы поддерживать производительность на хорошем уровне, организации могут развивать культуру психологической безопасности и делать разумные инвестиции в инструменты, поиск информации и сокращение технического долга, создавая гибкие и расширяемые системы визуализации и мониторинга.

**Существует правильный способ управлять процессом согласования изменений, который ведет к повышению скорости, стабильности и снижает риск профессионального выгорания**

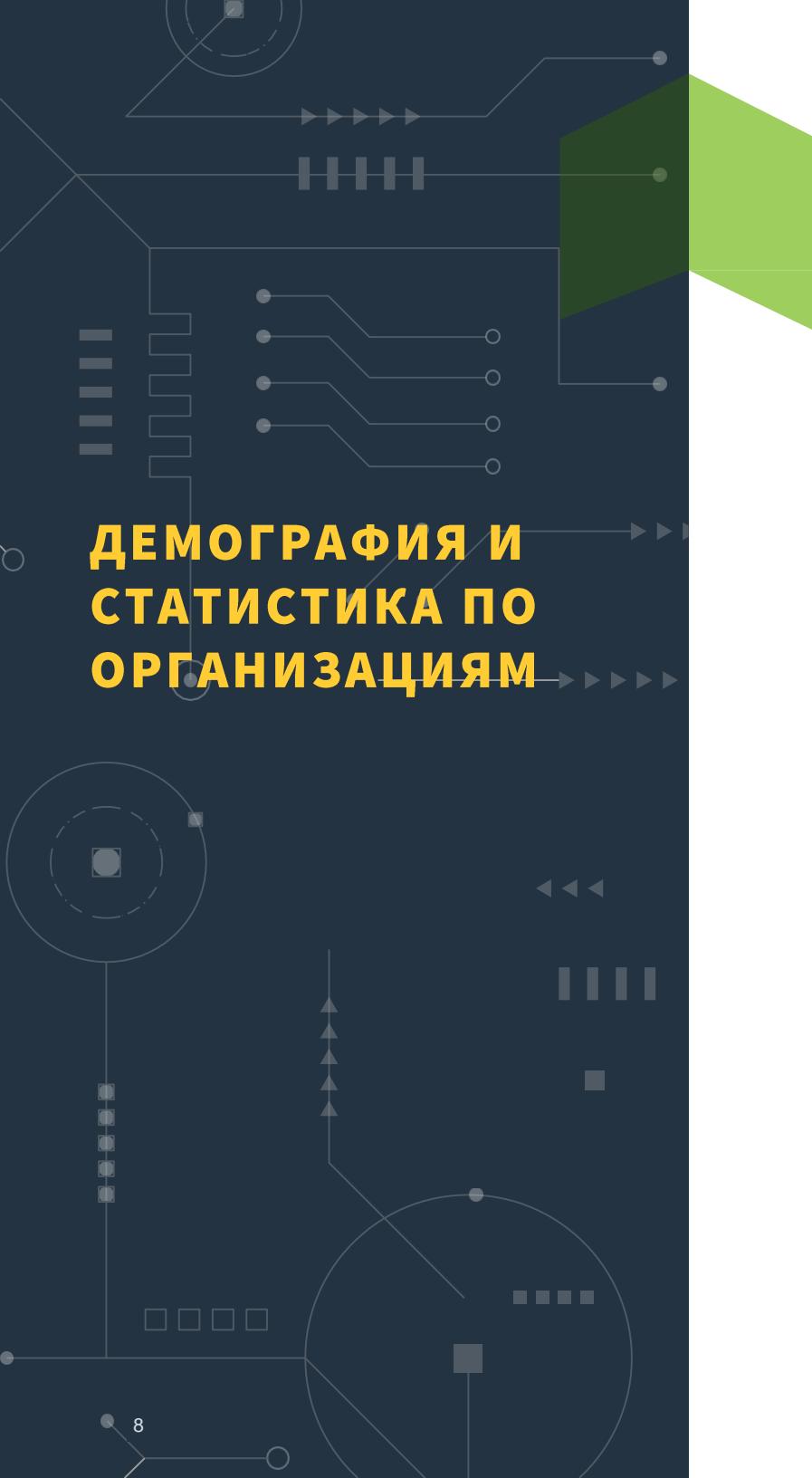
Громоздкие процедуры согласования изменений, требующие рассмотрения специальными комиссиями, отрицательно влияют на скорость и стабильность. И наоборот, наличие прозрачного и понятного процесса управления изменениями позволяет согласовывать изменения быстрее и более предсказуемо, а также снижает риск профессионального выгорания.

# КТО УЧАСТВОВАЛ В ИССЛЕДОВАНИИ?



Исследование DORA рассматривает практики, применяемые в отрасли разработки программного обеспечения, и практики DevOps. В своих выводах мы опираемся на данные, полученные на основе проведенных в течение шести лет научных исследований, в ходе которых были опрошены более 31 000 респондентов — практикующих экспертов из различных компаний. В 2019 году отчет пополнился данными опросов еще почти 1000<sup>2</sup> респондентов, представляющих различные отрасли по всему миру. По сравнению с прошлым годом в целом состав участников по основным демографическим и организационными категориям остался прежним. Единственное заметное различие, которое мы обнаружили, — это сокращение доли женщин в командах продуктовой разработки программного обеспечения.

<sup>2</sup> В исследовании приняли участие почти 1000 респондентов; погрешность анализа составляет 3 %, исходя из предположения о том, что во всем мире разработкой программного обеспечения заняты 23 миллиона экспертов, а доверительный интервал — 95 %.



## ДЕМОГРАФИЯ И СТАТИСТИКА ПО ОРГАНИЗАЦИЯМ

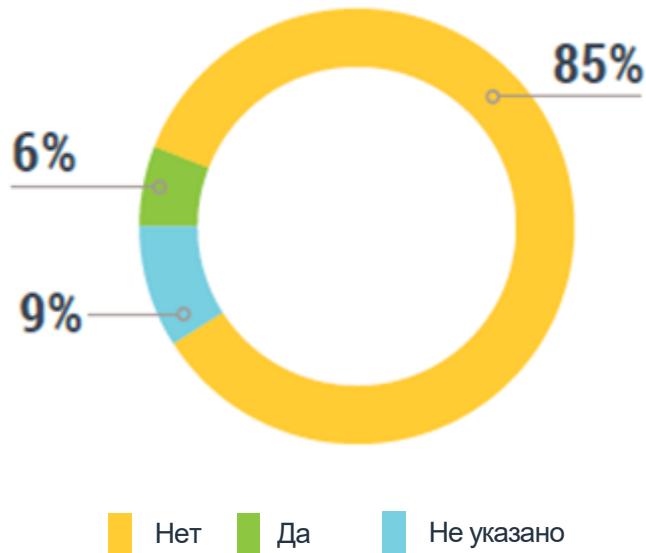
По основным демографическим категориям (гендерные, группы с ограниченными возможностями и недостаточно представленные группы) состав респондентов идентичен прошлому году. В целом гендерный состав респондентов остался тем же. Отчет показал снижение процента женщин в командах по сравнению с прошлым годом.

Также сохранилась репрезентативность основных организационных категорий: размер компании, отрасль и регион. Большая часть респондентов — это инженеры и руководители из технологических отраслей. По-прежнему представлены сотрудники различных направлений, такие как консультанты, коучи, специалисты по продажам и маркетингу. Кроме того, в состав респондентов продолжают входить организации из строго регулируемых отраслей, таких как финансовые услуги, государственный сектор, здравоохранение и розничная торговля.

## ГЕНДЕРНЫЙ СОСТАВ

По данным опроса этого года, гендерный состав остался прежним: 83 % мужчин (83 % в прошлом году), 10 % женщин (12 % в прошлом году) и <1 % лиц третьего пола (<1 % в прошлом году).<sup>3</sup>

*Ответы респондентов в этом году показали, что женщины входят в состав лишь 16 % команд (медиана). Это значение снизилось по сравнению с 25 % в прошлом году*



## ЛИЦА С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ

Ограничение возможностей определялось по шести измерениям согласно рекомендациям из краткого перечня вопросов Washington Group. Мы задаем вопрос об ограниченных возможностях второй год. В 2018 и 2019 гг. доля таких лиц составила 6 %.<sup>4</sup>

<sup>3</sup> Эти соотношения аналогичны полученным Stack Overflow в ходе опроса разработчиков 2019 года (90 % мужчин и 10 % женщин).

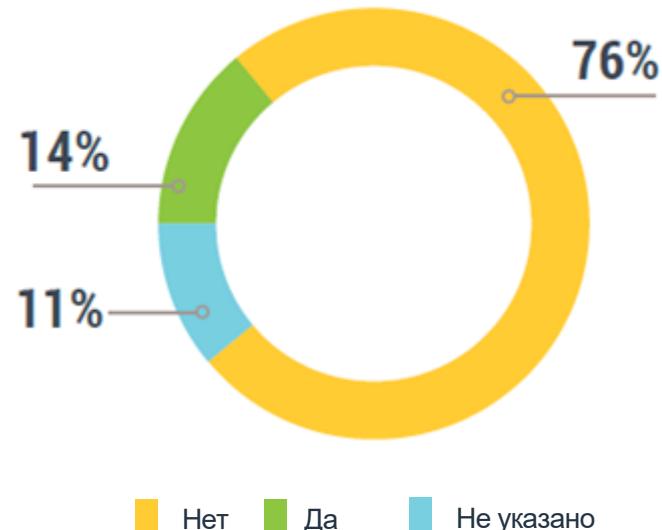
В этом опросе варианты «третий пол» и «пол не указан» не использовались. <https://insights.stackoverflow.com/survey/2019>

<sup>4</sup> Это соотношение соответствует показателям по отрасли; например, согласно опросу разработчиков, проведенному Stack Overflow в 2019 г., 6 % от общего числа респондентов являются лицами с ограниченными возможностями.

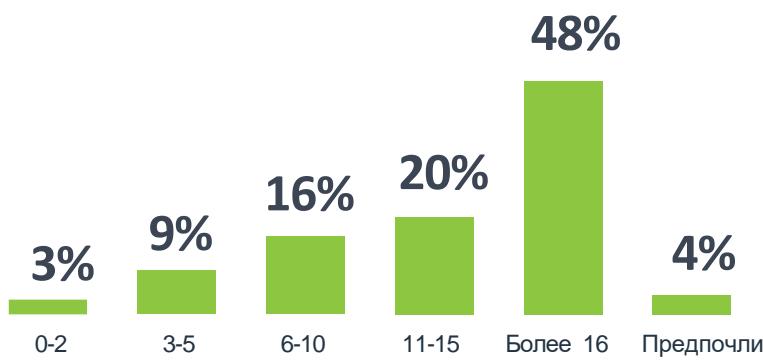
<https://insights.stackoverflow.com/survey/2019>

# НЕДОСТАТОЧНО ПРЕДСТАВЛЕННЫЕ ГРУППЫ

Принадлежность к недостаточно представленным группам определяется, например, по расовому, гендерному и другим признакам. Мы собираем эти данные третий год. Доля таких лиц остается относительно стабильной: 13 % в 2018 г. и 14 % в 2019 г.



\* > 100% из-за округления



## ОПЫТ РАБОТЫ

Аналогично прошлому году, у значительной доли респондентов опыт работы составляет более 16 лет (50 % в прошлом году), далее следуют специалисты с опытом работы 11–15 лет (20 % в прошлом году).

*Общий демографический состав в 2019 г. соответствует 2018 г. с небольшими процентными отличиями в рамках погрешности.*

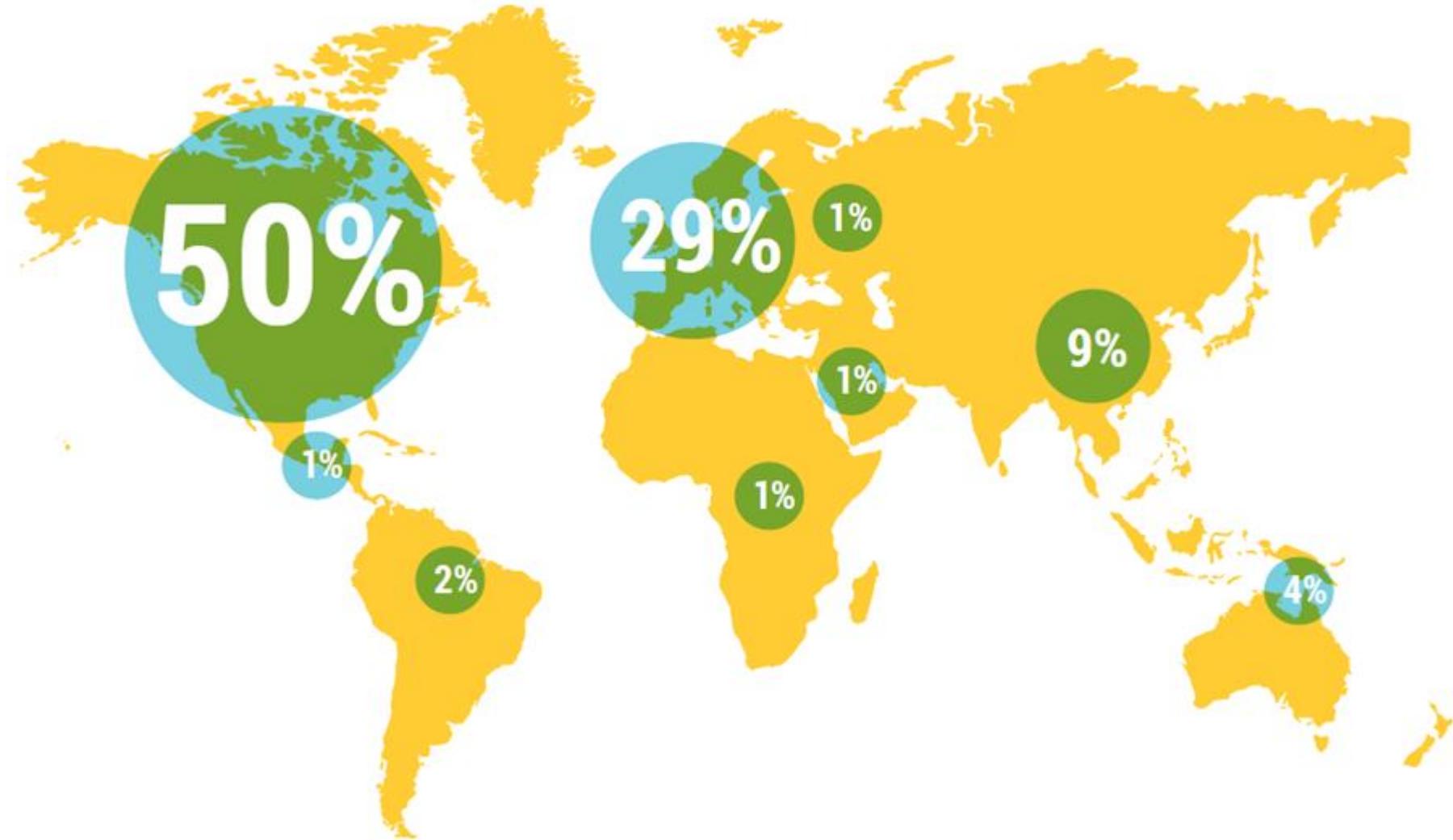
## ПОДРАЗДЕЛЕНИЯ

С начала исследования увеличилось количество участников, работающих в DevOps командах: 16 % в 2014 г., 19 % в 2015 г., 22 % в 2016 г. и 27 % в течение последних трех лет. (В пределах допустимой погрешности.)



## ОТРАСЛИ

Аналогично прошлому году, большинство респондентов работает в сфере высоких технологий, далее следуют финансовые услуги, розничная торговля и прочие отрасли.



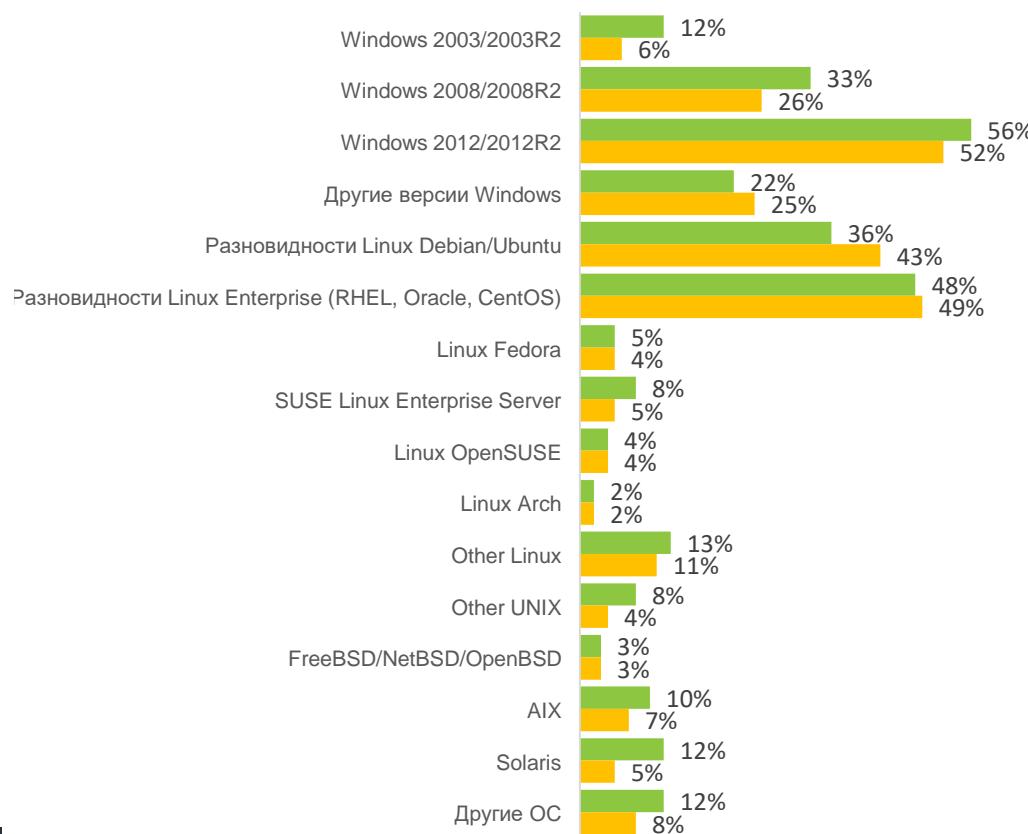
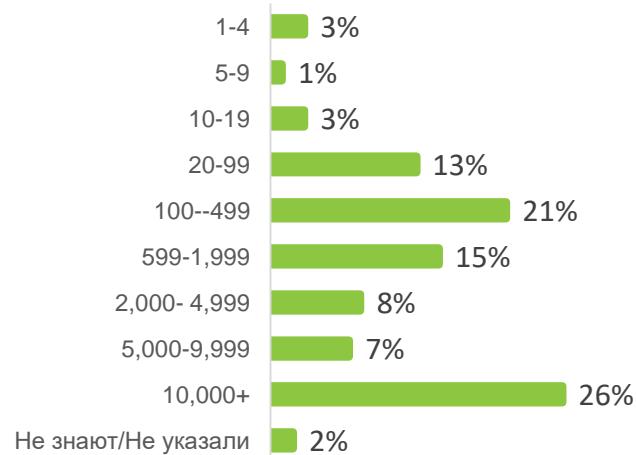
## РЕГИОНЫ

Аналогично прошлому году, около половины респондентов представляют компании из Северной Америки, далее следуют Европа/Великобритания — 29 %. Количество респондентов из Азии сократилось с 18 % в прошлом году до 9 % в этом.

## ЧИСЛЕННОСТЬ СОТРУДНИКОВ

Каждый четвертый респондент работает в очень крупной компании (более 10 000 сотрудников) — этот вариант выбран в 26 % ответов.

Каждый второй — в менее крупной (20–1999 сотрудников). Это распределение аналогично 2018 году, при этом доля компаний с числом сотрудников 500–1999 сократилась (на 12 % по сравнению с 2018 г.), а доля компаний с числом сотрудников 100–499 увеличилась (на 7 % по сравнению с 2018 г.).



## ОПЕРАЦИОННЫЕ СИСТЕМЫ

Распределение операционных систем также практически не изменилось с прошлого года.



# КАК ПРОВОДИЛОСЬ СРАВНЕНИЕ?



Этот раздел предназначен для сравнительной оценки эффективности DevOps в вашей компании. С помощью строгих статистических методов мы исследуем, каким образом команды разрабатывают, поставляют и эксплуатируют программное обеспечение. Эталонные значения для уровней «elite», «high», «medium» и «low performers» помогают определить, где вы находитесь в контексте показателей данного отчета. Здесь также обозначены тенденции от года к году.

# ЭФФЕКТИВНОСТЬ ПОСТАВКИ ПО И ОПЕРАЦИОННОЙ ДЕЯТЕЛЬНОСТИ

Для достижения своих целей организации все больше рассчитывают на способность поставлять и эксплуатировать программные решения. Чтобы сравнивать результаты по этому основному показателю, отрасли нужен способ измерения эффективности практик разработки и поставки программного обеспечения. За последние шесть лет мы разработали и проверили четыре показателя, дающих высокоуровневое систематическое представление о поставке и эффективности программного обеспечения, которые позволяют прогнозировать способность организации достигать поставленных целей. В прошлом году мы добавили дополнительную метрику, связанную с операционной деятельностью, и выяснили, что этот аспект помогает компаниям показывать отличные результаты. Мы называем эти пять параметров показателями эффективности поставки программного обеспечения и операционной деятельности. С их помощью мы можем рассматривать результаты на системном уровне. Такой подход позволяет при расчете метрик программного обеспечения избежать подводных камней, которые нередко ведут к разногласиям между функциональными подразделениями и позволяют достичь лишь локальной оптимизации в ущерб общему результату.

Первые четыре показателя, оценивающие эффективность процесса разработки и поставки, можно обобщить с точки зрения пропускной способности и стабильности. Пропускная способность процесса поставки программного обеспечения измеряется по срокам реализации изменений кода — от

коммита до поставки, а также по частоте развертываний. Стабильность измеряется по времени восстановления (от обнаружения инцидента, затрагивающего пользователей, до его устранения) и доле неудачных изменений, которая служит показателем качества процесса выпуска.

## МЕТРИКИ ПРОИЗВОДИТЕЛЬНОСТИ



### РАЗРАБОТКА ПО

Срок реализации



### РАЗВЕРТЫВАНИЕ ПО

Неудачные изменения



### РАБОТА СЕРВИСА

Доступность

Частота развертываний

Время восстановления

4 КЛЮЧЕВЫЕ МЕТРИКИ



Многие эксперты рассматривают эти метрики как набор компромиссных решений, полагая, что повышение пропускной способности отрицательно скажется на надежности процесса поставки программного обеспечения и на доступности сервисов. Тем не менее уже шестой год подряд наше исследование показывает, что скорость и стабильность — это взаимно усиливающие факторы. Кластерный анализ четырех показателей поставки программного обеспечения на основе данных 2019 года позволил выделить четыре профиля эффективности со статистически значимыми различиями по пропускной способности и стабильности<sup>5</sup>. Как и в прошлые годы, самые эффективные игроки показали лучшие результаты по всем четырем метрикам, а команды с низкой производительностью — значительно хуже.

Помимо скорости и стабильности, с точки зрения эффективности операционной деятельности важен такой показатель, как доступность. В общих чертах, доступность отражает способность технологических команд и организаций выполнять обещания в отношении программного обеспечения над которым они работают. В частности, под доступностью продукта или сервиса понимается возможность доступа к нему и его использования конечными пользователями<sup>6</sup>. Этот показатель отражает, насколько четко команды определили цели по доступности, отслеживают текущую доступность и делают выводы по итогам сбоев, обеспечивая замкнутость циклов обратной связи. Средства оценки доступности формируют достоверный и надежный измеряемый конструкт.

---

<sup>5</sup> Кластерный анализ не учитывает доступность, потому что ее показатели по-разному применяются к сервисам и программным решениям, предлагаемым в других форматах, например в виде пакетов программного обеспечения или прошивки.

<sup>6</sup> Задать цели по доступности команды могут с помощью соглашения об уровне обслуживания (SLA) и целей уровня обслуживания (SLO), а оценить свою эффективность — по показателям уровня обслуживания (SLI). Подробнее о разработке SLA, SLO и SLI читайте в книге Б. Бейера [Site Reliability Engineering. Надежность и безотказность как в Google](#)

Показатель эффективности поставки программного обеспечения*	Elite	High	Medium	Low
<b>Частота развертываний</b> По основному приложению или сервису над которым вы работаете. Как часто ваша организация развертывает код в промышленной среде или делает релизы для конечных пользователей?	По запросу (несколько развертываний в день)	От одного раза в день до одного раза в неделю	От одного раза в неделю до одного раза в месяц	От одного раза в месяц до одного раза в шесть месяцев
<b>Срок реализации изменений</b> По основному приложению или сервису над которым вы работаете. Каков срок реализации изменений (т. е. сколько времени проходит от фиксации кода до его успешного запуска в промышленной среде)?	Менее чем один день	От одного дня до одной недели	От одной недели до месяца	От одного до шести месяцев
<b>Время восстановления сервиса</b> По основному приложению или сервису над которым вы работаете. Сколько времени обычно занимает его восстановление после инцидента или дефекта, затрагивающего пользователей (например, в случае незапланированных простоеов или падения качества обслуживания)?	Менее чем за час	Менее одного дня <sup>a</sup>	Менее одного дня <sup>a</sup>	От одной недели до месяца
<b>Доля неудачных изменений</b> По основному приложению или сервису, над которым вы работаете, какой процент изменений в промышленной среде или версиях, предоставленных пользователям, привел к ухудшению уровня обслуживания (например, к падению качества или недоступности сервиса) и потребовал исправления (в виде пакета исправлений, патча, отката к предыдущей версии)?	0-15% <sup>b,c</sup>	0-15% <sup>b,d</sup>	0-15% <sup>c,d</sup>	46-60%

В отчете приведены медианы, т. к. распределение не является нормальным.

Если не указано иное, все различия статистически значимы на основе апостериорного анализа Тьюки.

a, b, c. Средние значимо отличаются на основе апостериорного анализа Тьюки; медианы не показывают различий из-за исходных распределений.

d. Средние не являются значимо различными на основе апостериорного анализа Тьюки.

\* Визуальное представление четырех показателей приведено в Приложении А.



## **ВЛИЯНИЕ ОТРАСЛИ И ОРГАНИЗАЦИИ НА ЭФФЕКТИВНОСТЬ ПОСТАВКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ОПЕРАЦИОННОЙ ДЕЯТЕЛЬНОСТИ**

Также подтвердился вывод прошлого года о том, что более высокое качество программного обеспечения идет рука об руку с более высокой доступностью. Анализ показал, что показатели доступности значимо коррелируют с профилями эффективности поставки программного обеспечения: команды уровня «elite» и «high» последовательно выигрывают по доступности (у «elite performers» в 1,7 раза чаще встречаются развитые практики обеспечения доступности).

### **Растет отраслевая динамика**

Многие аналитики заявляют, что отрасль «преодолела пропасть» с точки зрения DevOps и технологической трансформации, и наш анализ этого года подтверждает данные наблюдения. Динамика отрасли возрастает. В этих условиях с переходом к облачным технологиям, играющим роль топлива для преобразований, можно повышать и скорость и стабильность. Это еще раз подчеркивает значимость технологий, благодаря которым организации приносят прибыль своим акционерам и стейкхолдерам.

7 Также следует отметить, что ни одна из этих практик не является исключительно облачной.

Нами был проведен дополнительный анализ (с применением контрольных переменных), чтобы выяснить, влияет ли размер отрасли или организации на эффективность поставки программного обеспечения и операционной деятельности. Мы не нашли свидетельств влияния отрасли, за исключением розничной торговли, что говорит о том, что высокий уровень эффективности достижим для организаций всех типов и размеров, в том числе из строго регулируемых направлений, таких как финансовые услуги и государственный сектор. В розничной торговле компании получают определенный выигрыш в скорости и стабильности.

Мы нашли свидетельства того, что крупные компании (на которых работает более 5000 сотрудников) менее эффективны по сравнению со схожими компаниями, в которых трудится менее 5000 сотрудников. Вероятно, это связано с рядом факторов, свойственных крупным организациям. Самые заметные из них: тяжеловесные процессы и средства контроля, сильно связанные архитектуры, провоцирующие задержки и нестабильность. Мы призываем предприятия не воспринимать эти выводы как оправдание низкой эффективности, а признать, что достичь превосходства можно, инициировав программу непрерывных улучшений и ориентируясь на другие подобные организации, которым удалось достичь прогресса в этом направлении.

Мы сравнили доли каждой группы эффективности за 2018 и 2019 гг.



## ELITE PERFORMERS

Если сравнить команды уровня «elite performers» и «low performers», в пользу первых будут следующие показатели:



в **208**  
раз чаще  
развертывают ПО

в **106**  
раз меньше  
время от коммита  
до поставки



в **2,604**  
раз быстрее  
восстанавливаются  
после аварии

в **7**  
раз ниже  
доля неудачных  
изменений



■ Объем поставок

■ Стабильность

# ОБЪЕМ ПОСТАВОК

## Частота развертываний

По полученным данным, «elite performers» команды выполняют развертывания по требованию несколько раз в день, что совпадает с данными за прошлые несколько лет. В сравнении с ними, «low performers» выполняют развертывания от раза в месяц (12 раз в год) до одного раза в шесть месяцев (2 поставки в год), что говорит о снижении эффективности по сравнению с прошлым годом. Нормальное число развертываний в год находится в диапазоне от 1460 (рассчитано как четыре в день × 365 дней) у самых эффективных компаний до 7 у «low performers» (среднее между 12 и двумя развертываниями). Расширенный анализ этого показателя выявил, что «elite performers» команды делают поставки в 208 раз чаще, чем «low performers». Стоит отметить, что четыре развертывания в день — это консервативная оценка, если сравнивать с такими компаниями, как CapitalOne (до 50 развертываний в день по новым продуктам)<sup>8</sup> или Amazon, Google и Netflix, которые делают тысячи поставок в день (в совокупности по сотням сервисов в промышленных средах).

---

8 В 2017: <https://www.informationweek.com/devops/capital-one-devops-at-its-core/d/d-id/1330515>

## Срок реализации изменений

Аналогичным образом, у «elite performers» команд срок реализации изменений составляет менее дня. Этот показатель измеряется как время от коммита кода до его успешного развертывания в промышленной среде. Здесь эффективность немного снизилась по сравнению с прошлым годом, когда наиболее эффективные игроки сообщали о сроках поставки менее часа. В противоположность «elite performers» командам у «low performers» сроки реализации составляют от одного до шести месяцев. С учетом того что самим эффективным командам на реализацию изменений нужно 24 часа (консервативная оценка для ответа «менее дня»), а «low performers» — 2555 часов (среднее между 730 и 4380 часами, соответствующими одному и шести месяцам), «elite performers» команды поставляют изменения в 106 раз быстрее, чем «low performers».



# СТАБИЛЬНОСТЬ

## Время восстановления сервиса

У «elite performers» команд восстановление сервиса занимает менее часа, у «low performers» — от недели до месяца.

При расчетах мы использовали консервативные оценки интервалов времени: час для самых эффективных компаний и среднее между неделей (168 часов) и месяцем (5040 часов) для «low performers». На основе полученных цифр, у компаний «elite performers» сервис восстанавливается в 2604 раза быстрее, чем у «low performers». Как отмечалось ранее, по сравнению с предыдущим годом значение времени восстановления сервиса осталось прежним как для «elite performers» команд, так и для «low performers».

## Доля неудачных изменений

По полученным данным, у «elite performers» команд доля неудачных изменений составляет от 0 до 15 %, а у «low performers» — от 46 до 60 %.

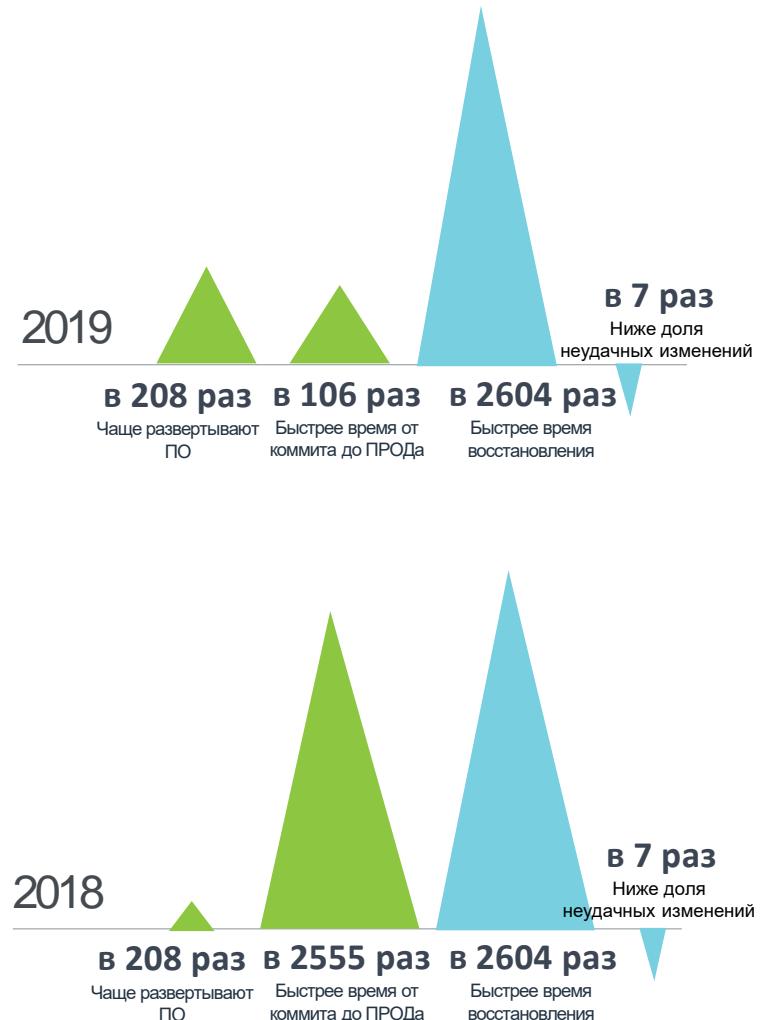
Среднее для этих двух диапазонов составляют 7,5 и 53 % соответственно. Это значит, что у «elite performers» команд доля неудачных изменений в семь раз меньше, чем у «low performers». Как отмечалось ранее, по сравнению с предыдущим годом значение доли неудачных изменений осталось прежним как для эффективных команд, так и для «low performers».

Все приведенные показатели являются относительными, т. е. представляют собой сравнение между категориями с лучшей и худшой эффективностью за каждый год. С 2018 по 2019 гг. разрыв между «elite performers» и «low performers» командами вырос или остался тем же по всем метрикам эффективности, кроме срока реализации изменений. Увеличение разрыва по частоте развертывания говорит о снижении эффективности у «low performers». Это может быть связано с возросшей сложностью окружений и сложностями в поставке программного обеспечения.

Сокращение различий между наименее и наиболее эффективными командами по срокам реализации изменений говорит о снижении эффективности у «elite performers» команд, для которых этот показатель составляет от часа до дня (раньше — менее часа). Это может отражать тенденцию применения более тяжеловесных процедур код-ревью и согласований, ставших популярными в последние годы.

## ПРОИЗВОДИТЕЛЬНОСТЬ ПОСТАВКИ ПО

Сравнение самых эффективных с самыми неэффективными.



# КАК ПОЛЬЗОВАТЬСЯ ИССЛЕДОВАТЕЛЬСКИМИ МОДЕЛЯМИ



Отчет этого года включает две исследовательские модели, призванные помочь в реализации инициатив по повышению эффективности и производительности. Почему же исследовательских моделей две? Чем они отличаются? В чем их сходство? И самое главное, как их использовать в качестве ориентира при принятии решений и выполнения рабочих задач?

## Начните с определения вашей цели...



Если требуется повысить эффективность поставки ПО и операционной деятельности, рассмотрите модель с этим конструктом и [перейдите в соответствующий раздел отчета](#), чтобы ознакомиться с рекомендациями по принципам, практикам и подходам, на которых нужно сосредоточить внимание.

Если требуется повысить производительность, рассмотрите модель с этим конструктом\* и перейдите в соответствующий раздел отчета, чтобы ознакомиться с указанием на чем нужно сосредоточить внимание.

## Использование моделей в качестве ориентира для трансформации

- Определите факторы, которые позволяют приблизиться к вашей цели (т. е. блоки, стрелки от которых ведут к составляющей, которую вы хотите улучшить). Как определено в данном отчете, это принципы, практики и подходы, которые потенциально стоит улучшить. (За пять лет исследований мы выявили дополнительные принципы, практики и подходы для эффективности поставки ПО и операционной деятельности.)<sup>9</sup>
- Помните, что для ускорения трансформации необходимо начать с основ и затем переключить внимание на содержащие факторы. Какие факторы вызывают самые большие задержки? Что вызывает основную головную боль? Где самые большие проблемы? Выберите от трех до пяти факторов и сразу выделите ресурсы на их устранение. Не волнуйтесь, если охвачены не все проблемы; сосредоточившись на самых крупных из них, вы устраниете узкие места, выявляете возможности для синергии и исключаете ненужную работу.

- Этой задачи есть и другие важные результаты. Совершенствование поставки ПО и операционной деятельности позволяют сократить количество случаев профессионального выгорания и проблем при развертывании (исследовано в 2016 и 2017 гг.), повысить уровень безопасности (исследовано в 2017 и 2018 гг.) и культуры (исследовано в 2014–2019 гг.). Дополнительные преимущества повышения производительности: улучшение баланса работы и личной жизни, сокращение количества случаев профессионального выгорания.

## Как читать исследовательские модели

Мы используем модель структурных уравнений (МСУ). Это прогнозная модель для тестирования взаимосвязей. Каждый блок соответствует элементу, измеряемому в нашем исследовании, а каждая стрелка — связи между элементами. Блоки большего размера, внутри которых расположены другие блоки (элементы), — это элементы второго порядка. Светло-голубой блок с пунктирной линией до другого элемента обозначает контрольную переменную. (См. полные модели на страницах [31](#) и [57](#).) Элементы, выделенные жирными буквами, исследовались в этом году впервые. Элементы в толстой рамке — это общие командные и организационные цели: Эффективность поставки ПО и операционной деятельности или производительность. Учитывайте это при определении своих целей и чтении моделей.

\* элемент или характеристика модели (примечание переводчика)

9 Вы можете найти все наши отчеты State of DevOps на [cloud.google.com/devops](http://cloud.google.com/devops)

При интерпретации моделей все линии со стрелками читаются как «прогнозирует», «затрагивает», «стимулирует» или «влияет». Например, элемент второго порядка «Эффективность поставки ПО и операционной деятельности» состоит из элементов «Эффективность поставки программного обеспечения» и «Доступность», которые в совокупности способствуют эффективности организации. Элемент «Тестирование аварийного восстановления» способствует обеспечению доступности. Элемент «Тестирование аварийного восстановления» выделен жирными буквами как впервые исследуемый в этом году. Линия со стрелкой на которой указан знак «→», обозначает отрицательное влияние элементов друг на друга; например, технический долг отрицательно влияет на (или снижает) производительность.

## Между двумя исследовательскими моделями есть некоторые пересечения

Это вызвано тем, что цели «Эффективность поставки ПО и операционной деятельности» и «Производительность» связаны во многих аспектах. В результате должны быть определены передовые методы создания и поставки технологий, представляющие ценность как для организаций, так и для отдельных лиц. Вполне разумно, что усилия по поддержке процессов доставки программного обеспечения положительно сказываются и на производительности тех, кто разрабатывает и поставляет программное обеспечение. Несмотря на схожесть, они по-прежнему измеряют разные результаты, поэтому мы анализируем их по-отдельности. Вот почему они представлены в двух разных исследовательских моделях.

## О чем говорят пересечения двух моделей

- Разумные инвестиции в повышение эффективности поставки ПО и операционной деятельности, как и в повышение производительности, могут сократить количество случаев профессионального выгорания. Это должно мотивировать как организации, так и инженерно-технических работников, так как требования к ним продолжают расти. По нашим наблюдениям, здоровый баланс между работой и личной жизнью является ключевым средством противодействия профессиональному выгоранию.
- Культура психологической безопасности вносит свой вклад в эффективность поставки ПО и операционной деятельности и производительность, подтверждая, что от развития и поощрения здоровой рабочей среды выигрывают и организации, и сотрудники.
- Усилия по обеспечению поддерживаемости кода, инвестиции в слабосвязанную архитектуру и средства мониторинга помогают повысить эффективность поставки ПО и операционной деятельности (за счет непрерывной поставки программного обеспечения) и производительность (за счет сокращения технического долга), подчеркивая важность хороших инструментов и систем.

# КАК ПОВЫСИТЬ ЭФФЕКТИВНОСТЬ ПОСТАВКИ ПО И ОПЕРАЦИОННОЙ ДЕЯТЕЛЬНОСТИ?



Основная цель цифровой трансформации — оптимизация поставки программного обеспечения: использование технологий для формирования ценности для клиентов и заинтересованных лиц. Наше исследование содержит проверенные на практике рекомендации, которые помогут вам сосредоточить внимание на принципах, практиках и подходах, важных для осуществления трансформации в вашей компании. Отчет этого года также содержит стратегии реализации, чтобы вы могли определить свой путь развития, который позволит добиться максимальных результатов.

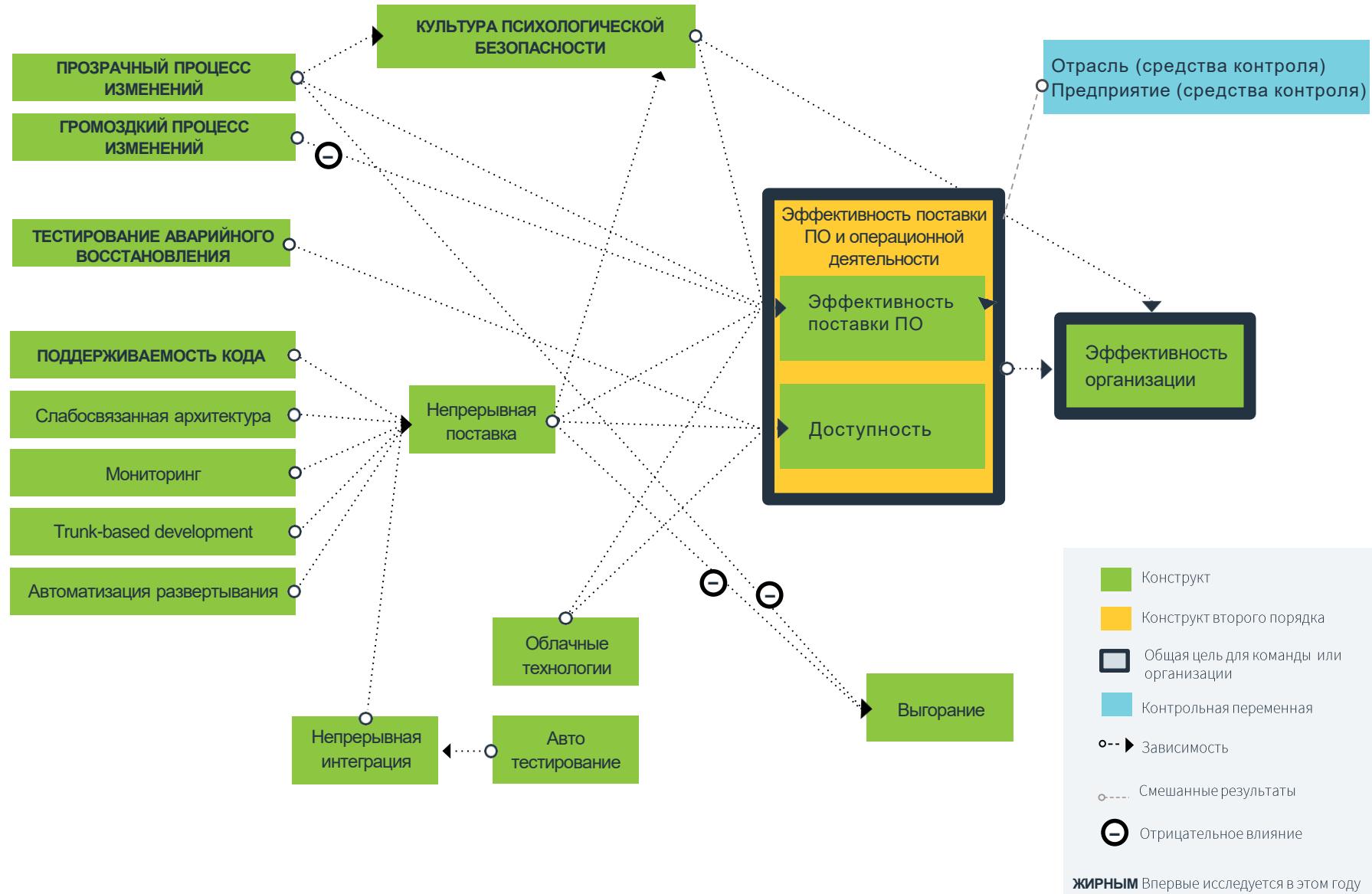


## ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПОСТАВКИ ПО И ОПЕРАЦИОННОЙ ДЕЯТЕЛЬНОСТИ

Изучите принципы, практики и подходы, описанные в нашем исследовании. Они послужат ориентиром для совершенствования процессов поставки технологий и формирования ценности. Начните с основ: базовых средств автоматизации (системы контроля версий и автоматизированного тестирования), мониторинга, понятных процедур согласования изменений и здоровой рабочей среды. Затем определите ограничивающие факторы, чтобы спланировать план развития. Эта стратегия подходит и тем, кто только приступает к трансформации, и тем, кто занимается оптимизацией уже не первый год. Сосредоточьте ресурсы на устраниении факторов, сдерживающих развитие компании в данный момент, затем повторите процесс снова: выявите ограничения и определите следующую цель.

Для обнаружения целевого показателя, требующего улучшения, и влияющих на него факторов [используйте модель на странице 31](#). Если ваша цель — повышение эффективности поставки программного обеспечения, такими факторами будут культура, прозрачный процесс реализации изменений, непрерывная поставка и использование облачных технологий. Затем сосредоточьтесь на самых значительных ограничениях.

# ЭФФЕКТИВНОСТЬ ПОСТАВКИ ПО И ОПЕРАЦИОННОЙ ДЕЯТЕЛЬНОСТИ



# СОВЕРШЕНСТВУЙСЯ ИЛИ УМРИ



Наш анализ не обнаружил свидетельств влияния отрасли на скорость и стабильность поставки программного обеспечения, за исключением розничной торговли, в которой наблюдалась более высокая эффективность поставки ПО и операционной деятельности. Это не удивительно с учетом исключительно высокой конкуренции в розничной торговле. В течение последних десяти лет по США прокатилась волна закрытий крупных магазинов, получившая название «ритейл-апокалипсиса». Выживают те, кому удается преуспеть с точки зрения рентабельности, производительности и удовлетворенности клиентов. Несспособность достичь превосходства в своей области приводит к уходу с рынка. Хотя розничные компании находятся на передовой изменений, вызванных высоким уровнем конкуренции, в плотную за ними следуют и другие отрасли, такие как финансовые услуги.

В этих условиях крайне важно успевать за темпами технологических изменений: организациям нужно удовлетворять и радовать требовательных клиентов и в то же время обеспечивать устойчивую выручку, устраивающую акционеров и стейкхолдеров. Розничная торговля — отличный пример отрасли, в которой технологии помогают получать прибыль. Компаниям из других сфер деятельности следует поучиться на этом опыте:

- Ретейлеры одними из первых стали использовать А/В-тестирование, чтобы понять покупательские привычки, предпочтения, взаимодействие клиентов с сайтами и приложениями и оптимизировать процесс покупки. Соответствующая техническая возможность потребовала более современных ИТ-решений и позволила создать эффективный цикл обратной связи для команд разработки продукта и маркетинга.

- Часто у розничных продавцов совсем небольшая маржинальность, поэтому для оперативного реагирования на изменения и готовности к масштабированию нужна эффективность и автоматизация.
- Ретейлеры должны уметь справляться с огромными колебаниями спроса либо столкнуться с риском потери бизнеса. Успех финансового года может зависеть от одной «черной пятницы». Розничные компании благодаря облачным технологиям легко наращивают свои мощности, а не тратят драгоценное время на обсуждения, нужно ли и когда следует подключать облачные ресурсы. Ресурсы им уже доступны.
- Ретейлеры научились ловко действовать в условиях строгого регулирования, потому что у них нет другого выхода. Пока другие отрасли могут позволить себе обвинять регуляторов в задержках внедрения современных технологий, конкуренция в рознице заставляет ретейлеров учиться быстро и безопасно работать в регулируемой среде. Именно так они и делают. В конце концов, сегодня уже сложно продавать товары, не поддерживая операции с использованием кредитных карт.

# ОБЛАЧНЫЕ РЕШЕНИЯ

По мере эволюции самой природы бизнеса все больше организаций выбирают мультиоблачные решения и гибридные облака. Это связано с тем, что помимо выигрыша в производительности, эти решения обеспечивают гибкость, контролируемость и доступность<sup>10</sup>. Ответы респондентов свидетельствуют о том, что по сравнению с прошлым годом мультиблаком и гибридными облаками стали пользоваться чаще.

Также мы просили респондентов указать, где размещено их основное приложение. Их ответы вновь показали, что пока нет единого мнения о том, что значит работать в гибридной среде или мультиблаке. Как мы отмечали в прошлогоднем отчете, различные респонденты могут определять гибридность по-своему. Если респонденты сообщают, что работают в гибридной среде, то мы принимаем их ответ как

есть. Когда эксперты пытаются обсуждать термины и состояние отрасли, часто возникает недопонимание (и существенно различающиеся отчеты отраслевых аналитиков): Мы не можем сравнивать то, что мы не можем определить и измерить.

## ХОСТИНГ ДЛЯ ОСНОВНОГО СЕРВИСА ИЛИ ПРИЛОЖЕНИЯ



10 [Transform Your Business with a Hybrid and Multicloud Strategy](#), Tilak, March 2019.



# СПОСОБ РЕАЛИЗАЦИИ ОБЛАЧНОЙ СРЕДЫ ИМЕЕТ ЗНАЧЕНИЕ

Если все по-разному понимают, что значит «в облаке», как же измерить преимущества? Мы обходим это ограничение, используя в качестве ориентира **основные характеристики облачных технологий\***, определенные Национальным институтом стандартов и технологии США (NIST). В рамках нашего опроса 80 % респондентов<sup>11</sup> сообщили, что основное поддерживаемое ими приложение или сервис размещены на облачной платформе. Используя фреймворк NIST, мы исследовали влияние основных практик на эффективность поставки ПО и операционной деятельности и второй год подряд выявили, что действительно важным является то как именно команды реализуют свои облачные сервисы, а непросто использования облачных технологий.

---

\* Термины, которые в этом отчете выделены жирным шрифтом, соответствуют конструктам исследовательских моделей на [страницах 31](#) и [57](#).

11 См. предыдущий график по способам размещения основного сервиса или приложения.

Обратите внимание, что можно было выбрать несколько вариантов ответа. Таким образом, ответы 80 % респондентов включали вариант «облачные технологии».

Облачные решения «elite performers» команд в 24 раза чаще, чем у «low performers», соответствуют всем основным характеристикам<sup>12</sup>. Это объясняет, почему команды и руководители, заявляющие о внедрении облачных технологий, также ощущают непонимание, так как не получают обещанного выигрыша в скорости и стабильности. Многие респонденты, указавшие, что используют облачные технологии, на самом деле не применили у себя важнейшие принципы. Только 29% респондентов, сообщивших об использовании облачной инфраструктуры, выбрали варианты «согласен» или «полностью согласен» в ответ на вопрос о соответствии всем пяти характеристикам, определенным NIST.

12 Эта цифра близка к результату прошлого года. Тогда «elite performers» команды в 23 раза опередили «low performers» по количеству ответов «согласен» или «полностью согласен» на вопрос о соответствии основным характеристикам облачных технологий.

## ПЯТЬ ОСНОВНЫХ ХАРАКТЕРИСТИК ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ





В прошлом году мы обнаружили, что «elite performers» команды чаще достигают соответствия всем пяти основным характеристикам облачных систем, что подтвердилось и в этом году. Эти характеристики важны при выборе средств реализации облачных технологий, поскольку позволяют разработать план действий для достижения успеха.

Исследование показало, что они влияют на эффективность поставки ПО и операционной деятельности.

Сосредоточившись на реализации в облаке — публичном, частном или гибридном — любая команда или организация может получить преимущества по скорости, стабильности и доступности.

---

<sup>13</sup> Термин «бессерверный» также используется для описания полнофункциональных клиентских приложений. Здесь он употребляется только в значении «функция как услуга» (FaaS). Подробнее об этом читайте в посте Майка Робертса (Mike Roberts): <https://martinfowler.com/articles/serverless.html#:~:text=What%20is%20Serverless>

## МАСШТАБИРОВАНИЕ ОБЛАЧНЫХ РЕШЕНИЙ



Очевидное преимущество облачных решений — масштабирование по запросу. Команды, которые используют преимущества динамического масштабирования, могут настроить свою инфраструктуру, так, что она будет гибко реагировать на запросы пользователей. За счет этого команды могут контролировать работу сервисов и автоматически масштабировать инфраструктуру по мере необходимости.

Используемая в облачных технологиях абстракция изменила то, как мы воспринимаем и представляем себе инфраструктуру. Для кого-то, кто использует оркестратор контейнеров, такой как Kubernetes или Mesos, пакет поставляется в виде образа контейнера с конфигурацией для развертывания. Типичные решения, предлагаемые по модели «платформа как услуга» (PaaS), больше склоняются к модели развертывания, сосредоточенной вокруг образов контейнеров как метода упаковки и сред выполнения контейнеров как инструмента исполнения. Мы видим это в таких продуктах, как Heroku, Google App Engine, Azure Container Instances, Cloud Foundry и Amazon Fargate. Бессерверные вычисления (также называемые «функция как услуга» или FaaS)<sup>13</sup> — это еще один шаг к упрощению развертывания, позволяющий потребителям заботиться только об исполнении кода приложения, абстрагируясь от масштабирования, планирования ресурсов и технического обслуживания от разработчиков и операторов. Примерами применения этой модели являются AWS Lambda, Azure Functions, GCP Cloud Functions и Zeit.

Со временем используемые в облачных вычислениях абстракции стали универсальными стандартами развертывания для поставщиков облачных ресурсов и платформ. Сеть, виртуальные машины, управление идентификацией и доступом (IAM), системы хранения данных и базы данных де-факто стали продуктами, которые предлагают все поставщики облачных служб в дополнение к возможностям машинного обучения, Интернета вещей (IoT), контейнерным решениям и языковым средам выполнения, а также средствам обеспечения безопасности. Мы продолжаем получать свидетельства того, что состояние продуктов сосредотачивается вокруг парадигм, связанных с контейнерами, языковыми средами выполнения и пакетами приложений.



## ЧЕРНЫЙ ЯЩИК ИЛИ ПРОЗРАЧНОСТЬ

# СТОИМОСТЬ ОБЛАЧНЫХ РЕШЕНИЙ

Облачные технологии также меняют наше понимание затрат на инфраструктуру и развертывание. ЦОД в целом или даже полная серверная стойка перестали быть единицей измерения. Клиенты поставщиков облачных сервисов платят только за то, что используют, располагая при этом возможностью масштабироваться при необходимости.

Помимо положительного влияния на эффективность поставки ПО и операционной деятельности, внедрение лучших практик облачных технологий помогает организациям увидеть полную картину расходов на эксплуатацию. Респондентам, реализовавшим все основные характеристики облачных решений, в 2,6 раза чаще удается точно оценить затраты на использование программного обеспечения. Также у них в два раза больше шансов легко определить свои наиболее дорогостоящие приложения и в 1,65 раза чаще оставаться в рамках операционного бюджета на ПО.

Почему при использовании облачных решений команды могут более точно оценивать и контролировать свои затраты? Скорее всего, причина в том, что облачные технологии позволяют получать точную информацию об использовании ресурсов инфраструктуры и расходах на разработчиков и экспертов по эксплуатации ИТ.

Прозрачность и осведомленность позволяют по-новому проектировать и строить системы, а также выстраивать программы поощрений. Хотя эта изменчивость сначала может вводить в заблуждение или перегружать тех, кто не привык к новой финансовой модели, команды могут выиграть за счет эффективной структуры, платя только за используемые вычислительные ресурсы.

И наоборот, центр обработки данных в традиционных средах часто воспринимается как «черный ящик», где очень сложно или невозможно получить информацию о стоимости обработки и цикла. Кроме того, природа капитальных затрат такова, что когда инфраструктура приобретена, эффективное проектирование уже не дает особого выигрыша. В этой связи капитальные расходы — это постоянные издержки, которые предсказуемы и понятны с самого начала. Однако о них редко известно группе проектирования, и их невозможно избежать даже при использовании более эффективных архитектур.

Финансисты могут сказать, что использование облачных технологий не привело к экономии в краткосрочной перспективе. Тем не менее мы знаем, что оно дает большую прозрачность информации. Как это возможно? В то время как поставщики облачных услуг прозрачно предоставляют информацию о затратах владельцам систем, пользователи не оплачивают эти расходы (кроме случаев применения модели взимания средств за использование или аналогичного механизма). Это может привести к сильным колебаниям расходов, из-за отсутствия контроля которых затраты на облачные технологии становятся непредсказуемыми. В таких сценариях команды, которые платят за инфраструктуру, могут предпочесть центры обработки данных как более прогнозируемый формат, даже несмотря на то, что отсутствие прозрачности затрат лишает пользователей стимула создавать более эффективные системы. Мы рекомендуем организациям выстраивать соответствующие программы поощрения, чтобы у владельцев системы были и данные, и мотивация для построения более эффективных систем. Здесь пригодится модель взимания средств за использование (чарджбэк) или аналогичный механизм.

В то время как некоторые решения больше подходят для локального развертывания (особенно те, от которых требуется надежность и предсказуемость), есть и другие преимущества использования облачной инфраструктуры, например возможность применения модели «инфраструктура как код» (Infrastructure-as-Code, IaC).

# ТЕХНИЧЕСКИЕ ПРАКТИКИ

## Действия для максимального эффекта

Многим организациям, желающим внедрить DevOps, в качестве ориентира нужен набор заранее определенных шагов или рекомендованных практик. Однако все организации разные, и выбор конкретных практик зависит от текущего состояния, в том числе от технологий, культуры и процессов, а также от краткосрочных и долгосрочных целей.

Здесь нужен комплексный подход, где первая задача — выявить ограничения текущего процесса поставки программного обеспечения с учетом краткосрочных и долгосрочных результатов, поддающихся измерению. Затем следует дать командам возможность решить, как эффективнее всего достичь этих результатов. В конце концов именно они эксперты в своих сферах работы и контексте.<sup>14</sup> Компании, применяющие этот подход, получают масштабируемые и гибкие решения. У их руководства нет необходимости скрупулезно контролировать детальные планы, оно может сосредоточиться на высокоуровневых результатах, позволяя своим организациям развиваться. Сосредоточившись на проектировании и достижении краткосрочных результатов, поддерживающих долгосрочную стратегию, команды адаптируются к внезапным и непредсказуемым проблемам. Кроме того, они опережают коллег, чьи трех и пятилетние планы не позволяют действовать столь гибко и подвижно, как того

14 Эта методика создана на основе работы Майка Ротера (Mike Rother) по исследованию компании Toyota. Она подробно описана на странице <http://www-personal.umich.edu/~mrother/Homepage.html>.

## ПРИНЦИПЫ, ПРАКТИКИ И ПОДХОДЫ, ИССЛЕДОВАННЫЕ В 2019 ГОДУ<sup>16</sup>

Организационный уровень	<ul style="list-style-type: none"><li>• Слабосвязанная архитектура</li><li>• Прозрачный процесс реализации изменений</li><li>• Поддерживаемость кода</li></ul>
Командный уровень	<ul style="list-style-type: none"><li>• Непрерывная интеграция</li><li>• Автоматизированное тестирование</li><li>• Автоматизация развертывания</li><li>• Мониторинг</li><li>• Trunk-based Development</li></ul>
Оба уровня: командный и организационный	<ul style="list-style-type: none"><li>• Использование облачных сервисов</li><li>• Тестирование аварийного восстановления</li></ul>

требуют изменения потребностей клиентов, технологический ландшафт и внезапные угрозы безопасности.<sup>15</sup>

Универсального способа улучшения не существует, поэтому в данном документе мы рассмотрели несколько аспектов, которые могут помочь организациям внедрить DevOps. Эти темы особенно актуальны для компаний, которым требуется ускорить трансформацию при наличии труднопреодолимых комплексных ограничений.

### Параллельные усилия на командном и организационном уровне

Некоторые принципы, практики и подходы развиваются на уровне команды, а другие (обычно в крупных компаниях или организациях со строгой иерархией) — на уровне организации в целом. Эти два потока — командный и организационный — могут и должны выполняться одновременно, поскольку зачастую они поддерживают друг друга.

Например, создание платформы непрерывной интеграции, которая упрощает получение быстрой обратной связи от автотестов, может значительно повысить результативность работы при использовании несколькими командами в организации.

15 Для успешного применения этой методики крайне важно регулярно предоставлять командам необходимые ресурсы и мощности.

16 Полный список принципов, практик и подходов, способствующих повышению эффективности поставки ПО и операционной деятельности, приведен в Приложении А книги *Accelerate: The Science of Lean Software and DevOps* (обобщение результатов отчетов State of DevOps 2014–2017 гг.), в материале *Accelerate State of DevOps (2018 г.)* и в данном отчете.

Аналогично эффект от автоматизации развертывания на уровне команды будет небольшим, если ее код внедряется только вместе с кодом других команд. Это подводит нас к архитектурному препятствию, которое необходимо устранять на организационном уровне (для чего, скорее всего, потребуются усилия отдельных команд).

Мы рассмотрим эти принципы, практики и подходы подробнее, изучив их сквозь призму командного и организационного уровней.

Помните, что наша цель — улучшение способности компании поставлять программное обеспечение. Для ее достижения используются технические практики в области поставки и развертывания, которые мы называем **непрерывной поставкой** (*continuous delivery, CD*). Непрерывная поставка снижает риски и затраты, связанные с выпуском релизов. Тем не менее, если ваша организация хочет стать успешной, недостаточно внедрить непрерывную поставку ради непрерывной поставки. К ее реализации следует подходить с учетом таких организационных целей, как рентабельность, производительность и удовлетворенность клиентов.

## КАК МЫ ИЗМЕРЯЕМ НЕПРЕРЫВНУЮ ПОСТАВКУ

Команды могут выполнять развертывания в промышленной среде или для конечных пользователей по запросу на протяжении всего цикла поставки программного обеспечения.

Каждый член команды имеет доступ к оперативной обратной связи по качеству и удобству развертывания системы, а принятие мер в соответствии с полученными таким образом данными — главный приоритет для команды.

## Технические принципы, практики и подходы командного уровня

В предыдущие годы мы обнаружили, что автоматизация тестирования благоприятно влияла на непрерывную поставку (CD). В этом году мы продолжили это исследование и выяснили, что автоматизация тестирования положительно влияет и на непрерывную интеграцию (continuous integration, CI). Автотесты дают разработчикам уверенность в том, что ошибки при выполнении набора тестов соответствуют ошибкам на реальном окружении, а успешное прохождение набора тестов действительно означает, что приложение можно развертывать. Способность воспроизводить и исправлять ошибки, собирать обратную связь по тестам,

*В чем отличия от предыдущего исследования, и что это значит для вас?*

*В предыдущие годы мы просто проверяли важность автоматизированного тестирования и непрерывной интеграции (CI), но не рассматривали их взаимодействие. В этом году мы обнаружили, что наличие автотестов способствует улучшениям в CI. Это значит, что разумные инвестиции в разработку наборов автотестов помогут усовершенствовать CI.*

повышать качество тестов и быстро повторять прогоны теста также имеет отношение к автоматизированному тестированию.

Мы подтвердили данные о том, что CI улучшает CD. Чтобы непрерывная интеграция давала результат, каждый коммит кода должен сопровождаться созданием сборки программного обеспечения и выполнением определенных наборов тестов. Автоматизированные сборки и тесты для проекта должны успешно выполняться на повседневной основе.

Также подтвердилось, что на непрерывную поставку влияют: автоматизация развертывания, разработка на основе главной ветки (trunk-based development, TBD)<sup>17</sup> и мониторинг. У этих принципов, практик и подходов могут быть зависимости от организационного уровня, аналогичные описанным нами для автоматизации развертывания. Например, команда может контролировать собственный код, но не получит от этого всех преимуществ, если мониторинг приложений и инфраструктуры отсутствует и не используется при принятии решений.

<sup>17</sup> Наше исследование показало, что для эффективной TBD разработки характерно использование менее трех активных веток, при этом ветки и ветвления (fork) «живут» менее дня до вливания в основную ветку (master).

## Технические принципы, практики и подходы организационного уровня

В отличие от принципов, практик и подходов, которые можно внедрить на командном уровне, получив быструю отдачу, некоторые принципы, практики и подходы гораздо эффективнее при координации и поддержке на организационном уровне. К последним относятся те, что требуют принятия решений или проектирования с участием нескольких команд, например создание архитектуры или политики (например, управления изменениями).

Исследование этого года подтвердило положительное влияние **слабосвязанной архитектуры** на непрерывную поставку.

Слабосвязанная архитектура предполагает, что выполняющие поставку команды могут независимо тестировать, развертывать и изменять свои системы по запросу. Они не зависят от других команд с точки зрения дополнительной поддержки, услуг, ресурсов или согласований, за счет чего сокращается объем коммуникаций для принятия решения. При таком подходе команды могут быстро создавать ценность, однако для этого требуется координация на более высоком уровне.

## РАЗРАБОТКА ПО С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

Наше исследование сосредоточено на разработке и поставке программного обеспечения в организационном контексте, когда команды полностью заняты разработкой и поставкой приложений, что позволяет их участникам координировать разработку и релизы с гораздо более плотным графиком выпусков. Мы обнаружили, что разработка на основе TBD с частыми коммитами в основную ветку и с развертыванием в производственной среде позволяет получить предсказуемую эффективную отдачу.

А как обстоит дело с разработкой программного обеспечения с открытым исходным кодом?

У проектов с открытым исходным кодом другой набор сроков и ожиданий, т. к. они по большей части развиваются сообществом. Участники из разных уголков мира отправляют свои патчи в проекты, когда им позволяет график. В связи с тем, что в проектах с открытым исходным кодом важно обеспечить взаимодействие между людьми из разных стран и организаций (включая фрилансеров, энтузиастов и разработчиков всех уровней), релизы в таких проектах существенно отличаются по стилю от практики непрерывной поставки программного обеспечения. Обычно они срезаются с ветки в определенный момент после серьезного тестирования.

По некоторым аспектам результаты нашего исследования применимы и к разработке с открытым исходным кодом.

- Чем раньше коммитится код, тем лучше. В проектах разработки программного обеспечения с открытым исходным кодом многие наблюдали, что быстрое вливание патчей с целью предотвращения перемещений (*rebase*) помогает разработчикам работать быстрее.
- Лучше работать с небольшими изменениями кода. Крупные наборы исправлений (*patch bombs*) труднее и медленнее вливаются в проект, чем более мелкие и читаемые, так как в первом случае специалистам поддержки требуется больше времени на проверку изменений.

И при работе с базой закрытого исходного кода, и в проектах с открытым исходным кодом ветки с коротким жизненным циклом, небольшие удобочитаемые патчи и автоматизированное тестирование изменений позволяют каждому быть более продуктивным.



Архитектурные подходы, лежащие в основе этой стратегии, включают использование ограниченных контекстов (bounded context) и API как способа разъединить крупные предметные области на небольшие блоки, слабо связанные между собой. Этому должны способствовать как сервис-ориентированные архитектуры, так и любые по-настоящему микросервисные архитектуры<sup>18</sup>. Архитектура, позволяющая независимо проводить тестирование и развертывание сервисов, помогает командам достичь большей производительности.

Чтобы наши системы работали, нужно много кода. В Facebook 62 миллиона строк кода (не включая серверную часть), в операционной системе Android — от 12 до 15 миллионов строк кода, а в типичном приложении для iPhone — 50 тысяч строк кода<sup>19</sup>. Учитывая, сколь огромен объем кода, необходимый для работы наших компаний, мы хотели исследовать, какие связанные с кодом практики действительно помогают повысить эффективность (если помогают). Мы называем

это «поддерживаемостью кода».

Наш анализ показал, что поддерживаемость кода положительно влияет на непрерывную поставку. Команды, хорошо контролирующие поддерживаемость кода, имеют системы и инструменты, которые упрощают для разработчиков процесс изменения кода, поддерживаемого другими командами, поиск примеров в базе исходного кода, повторное использование кода, написанного другими, а также добавление, обновление и переход на новые версии зависимостей без переписывания существующего кода.

Наличие этих систем и инструментов не только поддерживает непрерывную поставку, но и помогает сокращать технический долг, за счет чего повышается производительность, о которой пойдет речь в следующем разделе.

Организации, тратящие время на поддерживаемость кода, дают реальные преимущества своим инженерам.

---

<sup>18</sup> Важно не допустить преждевременной декомпозиции новых систем на сервисы или слишком мелкомодульные сервисы, чтобы не допустить снижения эффективности поставки. Мартин Фоулер (Martin Fowler) освещает этот вопрос в своей публикации *MonolithFirst* <https://martinfowler.com/bliki/MonolithFirst.html>.

<sup>19</sup> <https://www.businessinsider.com/how-many-lines-of-code-it-takes-to-run-different-software-2017-2>



К примеру, управлять зависимостями совсем непросто. Обновление зависимости может открыть «кроличью нору» к таким проблемам, как критическое изменение API, обновление транзитивной зависимости, создание несовместимых зависимостей (например, проблема ромбической зависимости, diamond dependency issue) и нарушение функциональности. Инструменты, которые позволяют избежать этих ошибок и пролить свет на последствия изменения кода, дадут инженерам возможность принимать более эффективные решения и выдавать более качественный код.

Легко пуститься в спор об инструментах и способах организации кода. Но важно сосредоточить внимание на результатах: Создаем ли мы условия или препятствия для обеспечения эффективности и производительности программного обеспечения?

## ПОЛЕЗНЫЕ И ПРОСТОЕ В ИСПОЛЬЗОВАНИИ ИНСТРУМЕНТЫ ДЛЯ РАЗВЕРТЫВАНИЯ ПО

Раньше мнение продвинутых пользователей (разработчиков, тестировщиков и системных администраторов) часто игнорировалось при рассмотрении такого вопроса, как удобство использования их инструментов. Иногда менеджмент, имеющий к технологиям весьма опосредованное отношение, предполагает, что технические специалисты научатся работать с любым предоставленным им инструментом. И такой образ мысли встречается довольно часто. Во время Второй мировой войны отбор и обучение пилотов были построены на их способности работать с чрезмерно сложными приборными панелями. Затем эксперты по юзабилити выяснили, что такая работа, как пилотирование самолета, сама по себе является сложной. Лучше спроектировать более простую в использовании и понятную приборную панель, чтобы пилоты могли уделять все внимание безопасному управлению воздушным судном.

В некоторых случаях юзабилити потребности игнорируются, поскольку руководство предполагает, что потребности технических специалистов и обычных конечных пользователей аналогичны. Сейчас нам известно, что опытные пользователи (такие как инженеры) часто имеют особые сценарии использования с уникальными потребностями в дизайне. Кроме того, технические специалисты владеют гораздо более широким набором навыков и опыта (интерфейсы, информационная безопасность, проектирование баз данных). Изначальное проектирование инструментов доступными и удобными в использовании — важный аспект для производителей таких инструментов.

В связи с этим в исследовании этого года мы изучили вопрос удобства использования инструментов, применяемых в поставке программного обеспечения. Практики, поддерживающие разработку и поставку программного обеспечения, важны с точки зрения скорости и стабильности.

Полезность и простота использования инструментов для развертывания тесно связаны с непрерывной интеграцией и поставкой. Это не лишено смысла, ведь чем больше инструменты подходят для выполняемой задачи, тем лучше можно ее выполнить.

Также мы выяснили, что удобство использования способствует повышению производительности, о чём вы можете прочитать на [странице 55](#).

# ТЕСТИРОВАНИЕ АВАРИЙНОГО ВОССТАНОВЛЕНИЯ

Каждая организация, у которой есть критичное для ее работы (mission-critical) программное обеспечение, должна иметь план аварийного восстановления. Однако создание таких планов без тестирования напоминает создание резервных копий без регулярных проверок возможности восстановления, т. е. оно бесполезно. Мы опросили респондентов, какие виды тестирования аварийного восстановления выполняются в их организациях.

## ТИПЫ ТЕСТИРОВАНИЯ АВАРИЙНОГО ВОССТАНОВЛЕНИЯ ПО ПРОФИЛЯМ ЭФФЕКТИВНОСТИ

	Low	Medium	High	Elite	ОБЩЕЕ
«Настольные учения», которые не проводятся на реальных системах	<b>35%</b>	<b>26%</b>	<b>27%</b>	<b>30%</b>	<b>28%</b>
Отказоустойчивость инфраструктуры (включая центр обработки данных)	<b>27%</b>	<b>43%</b>	<b>34%</b>	<b>38%</b>	<b>38%</b>
Отказоустойчивость приложений	<b>25%</b>	<b>46%</b>	<b>41%</b>	<b>49%</b>	<b>43%</b>
Моделирование, прерывающее работу тестовых систем, имитирующих промышленные (включая моделирование отказа, такого как ухудшение характеристик сетевых подключений, отключение маршрутизаторов и т. п.)	<b>18%</b>	<b>22%</b>	<b>23%</b>	<b>29%</b>	<b>23%</b>
Моделирование, нарушающее работу промышленных систем (включая моделирование отказа, такого как ухудшение характеристик сетевых подключений, отключение маршрутизаторов и т. п.)	<b>18%</b>	<b>11%</b>	<b>12%</b>	<b>13%</b>	<b>12%</b>
Создание средств и систем автоматизации, прерывающих работу промышленных систем на регулярной основе	<b>9%</b>	<b>8%</b>	<b>7%</b>	<b>9%</b>	<b>8%</b>

Не все тесты выполняются в промышленных системах, что обусловлено двумя причинами. Во-первых, создать полноценные копии промышленных систем трудно (и зачастую слишком дорого). Во-вторых, инциденты, выводящие из строя промышленные системы, часто возникают из-за взаимодействия компонентов, которые на первый взгляд работают normally, поэтому такие инциденты трудно встретить в тестовых средах. По мере усложнения систем эти факторы приобретают все большую значимость.

Мы спросили, с какой периодичностью организации проводят тестирование аварийного восстановления в промышленной инфраструктуре.

- Моделирование, нарушающее работу промышленных систем (включая моделирование отказа, такого как ухудшение характеристик сетевых подключений, отключение маршрутизаторов и т. п.)
- Отказоустойчивость инфраструктуры (включая центр обработки данных)
- Отказоустойчивость приложений

Только 40 % респондентов выполняют тестирование аварийного восстановления хотя бы раз в год, используя один или несколько указанных методов. Организации, которые проводят тесты аварийного восстановления, чаще имеют более высокий уровень доступности сервисов. Соответственно их технические команды способны давать и держать обещания, а также делать соответствующие заявления о программном продукте или сервисе, с которым они работают.



*Комментирует Майк Гарсия (Mike Garcia), вице-президент по обеспечению стабильности и надежности в Capital One: «Недостаточно продемонстрировать, что вы можете быстро делать поставки используя современные облачные технологии. В таких строго регулируемых отраслях, как банковская, у нас есть обязательства, которые требуют предоставить подтверждение уровня надежности, т. е. способности выполнять обязательства по обслуживанию клиентов. ... Для этого нам нужно не просто показать, что у нас есть отказоустойчивость в целом... Идея такова, чтобы постоянно показывать все более продвинутые практики и подходы и автоматическое обеспечение устойчивости с использованием более комплексных методик хаос тестирования».*



Организациям, которые для проведения проверок аварийного восстановления взаимодействуют друг с другом и с функциональными подразделениями, удается улучшить не только свои системы. В связи с тем, что эти тесты объединяют столько команд, они охватывают соединения, о которых многие уже забыли или не знали вовсе. Подобные упражнения также помогают совершенствовать и укреплять процессы и коммуникацию вокруг тестируемых систем, делая их более эффективными и результативными.

Также мы оценили, принимают ли организации какие-либо меры по результатам тестирования аварийного восстановления. Анализ показал, что организации, которые составляют и выполняют план мероприятий на основе выводов, полученных в результате тестирования аварийного восстановления, по категории эффективности в 1,4 раза чаще относятся к «elite performers».

## ВЫВОДЫ ПО РЕЗУЛЬТАТАМ ПРОВЕРОК АВАРИЙНОГО ВОССТАНОВЛЕНИЯ

Объективный послеаварийный анализ — это важный аспект поддержки развития и возможность сделать выводы по итогам отказа. Этот факт хорошо освещается в литературе и подтверждается в нашем предыдущем исследовании, которое показало, что проведение объективного послеаварийного анализа помогает сформировать культуру обучения и организационную культуру, которые будут оптимальны для повышения производительности программного обеспечения и эффективности компаний.

В качестве отличного материала по теме аварийного восстановления, где представлены обе точки зрения как на преимущества, так и на затраты от работающего инженера по обеспечению надежности, рекомендуем публикацию Крипы Кришнана (Kripa Krishnan) и Тома Лимончелли (Tom Limoncelli) *Weathering the Unexpected*.<sup>20</sup>

В этой статье на ACM Queuee Крипа Кришнан, директор в Google, ранее отвечающий за тестирование аварийного восстановления (DiRT), делает следующее наблюдение: «Чтобы DiRT мероприятия были успешными, организации нужно начать воспринимать отказы систем и процессов как возможность чему-то научиться. Что-то обязательно пойдет неправильно. Когда это происходит, необходимо сосредоточиться на исправлении ошибки, а не обвинять сотрудников или команды в отказе сложной системы».

---

<sup>20</sup> <https://queue.acm.org/detail.cfm?id=2371516>

# УПРАВЛЕНИЕ ИЗМЕНЕНИЯМИ

Внесение изменений в программное обеспечение и промышленные системы — нередко сложный и бюрократичный процесс. Большая часть этой сложности продиктована двумя факторами: потребностью в координации между командами и требованиями регулирующих органов, особенно в финансовом и государственном секторах, а также в здравоохранении. Хотя сложности, связанные с удовлетворением требований регуляторов, находятся вне зоны влияния руководства и специалистов, мы можем контролировать роль командной координации в процессе управления изменениями — и эта роль меняется.

Взять, к примеру, положение о разделении полномочий. Согласно ему, изменения должны утверждаться лицом, которое не является их автором, и это требование часто включается в нормативную базу<sup>21</sup>. Мы согласны, что один сотрудник не должен контролировать весь процесс от и до (в этом и смысл такого контроля), однако существуют более легкие и надежные способы решения этой задачи, не требующие такого объема координации, как существующие громоздкие подходы.

---

<sup>21</sup> Например, Стандарт безопасности данных индустрии платежных карт (PCI-DSS) и разработанная NIST структура управления рисками, используемая органами Федерального правительства США.



## ВНЕДРЕНИЕ РАЗДЕЛЕНИЯ ОБЯЗАННОСТЕЙ

Один из подходов требует, чтобы каждое изменение утверждалось отличным от автора членом команды в рамках код-ревью до коммита в систему контроля версий (в рамках парного программирования) или до слияния в основную ветку.

Такой подход можно сочетать с автоматизированными проверками, которые ограничивают изменения кода. Например, можно ввести проверки, не позволяющие разработчикам отправлять изменение (даже после проверки коллегой), если оно приведет к росту расходов на вычислительные ресурсы и хранение данных выше определенного порогового значения. Этот простой и понятный с точки зрения реализации процесс — простая возможность для специалистов улучшить процесс управления изменениями. Некоторые его сторонники, и в этом их поддерживает фреймворк управления ИТ-услугами (ITSM), заявляют, что формализованные процессы управления изменениями обеспечивают большую стабильность, и указывают, что они были отраслевой нормой на протяжении десятков лет.

При использовании код-ревью для разделения обязанностей необходима фиксация всех изменений промышленных систем в системе управления изменениями с указанием самого изменения и одного или нескольких инициировавших его сотрудников, а также логов процесса согласования. Если вы используете систему контроля версий в качестве единого источника достоверной информации по всем изменениям в системах (подход из парадигмы, известной как «инфраструктура как код» или gitops), то вам просто нужна возможность фиксировать согласующих каждое изменению. У этого подхода есть дополнительное преимущество: он делает процесс проверки и фиксации кода более удобным для аудита. Наше предыдущее исследование также показало, что полноценное использование системы контроля версий, в том числе для хранения конфигураций системы и скриптов, повышает эффективность.

Сторонники методологий бережливого производства (Lean) и гибкой разработки (Agile) могут спорить, что оптимизированные процессы согласования изменений позволяют быстрее получать обратную связь, улучшить информационные потоки и достигать более высоких результатов. Чтобы исследовать эти гипотезы, мы ввели два новых конструкта: один соответствует легкому и понятному процессу согласования изменений, а другой — более формальному и громоздкому, и проверили их влияние на эффективность поставки программного обеспечения.

## Громоздкий процесс реализации изменений

Мы обнаружили, что формализованные процессы управления изменениями, в рамках которых для изменения требуется согласование внешним органом, таким как совет по изменениям (change advisory board, CAB), или руководителем высшего звена, отрицательно влияют на эффективность поставки программного обеспечения. Участники опроса, ответившие, что в их организации используется подобный формальный процесс согласования, в 2,6 раза чаще оказывались «low performers» по эффективности. Это расширяет

выводы предыдущего исследования, в котором мы выявили отрицательную корреляцию между громоздким процессом согласования и долей неудачных изменений.<sup>22</sup>

Смысл громоздких процессов управления изменениями, предлагаемых фреймворками ITSM, состоит в снижении рисков, связанных с релизами. Чтобы разобраться в этом, мы проверили, связана ли более низкая доля неудачных изменений с наличием более формального процесса согласования изменений. Как и в прошлом исследовании, эта гипотеза не подтвердилась<sup>23</sup>. Также мы изучили, приводит ли введение дополнительных согласований к замедлению процесса и снижению частоты выпуска крупных релизов с сопутствующим ростом влияния на промышленные системы, которое скорее всего, связано с более высоким уровнем риска, а значит, и повышением доли неудачных изменений. Также мы изучили, приводит ли введение дополнительных согласований к замедлению процесса и снижению частоты выпуска крупных релизов с сопутствующим ростом влияния на промышленные системы, которое скорее всего, связано с более высоким уровнем риска, а значит, и повышением доли неудачных изменений.

22,23 Velasquez, N., Kim, G., Kersten, N., & Humble, J. (2014). Отчет State of DevOps: 2014. Puppet Labs.



## ЧТО ПРОИСХОДИТ С СОВЕТОМ ПО ИЗМЕНЕНИЯМ В ПАРАДИГМЕ НЕПРЕРЫВНОЙ ПОСТАВКИ?

Наша гипотеза была подтверждена полученными данными. Это имеет важные последствия для организаций, работающих над снижением рисков в процессе выпуска ПО. Организации часто реагируют на проблемы с релизами программного обеспечения, вводя дополнительные процессы и согласования. Наш анализ показал, что такой подход только ухудшает ситуацию.

Мы рекомендуем организациям отходить от внешнего согласования изменений из-за его отрицательного влияния на эффективность. Напротив, организациям стоит сделать «сдвиг влево» (shift left) к согласованию на основе проверок коллегами в ходе процесса разработки. Помимо этого, для более раннего обнаружения, предотвращения и исправления ошибочных изменений в цикле поставки следует использовать средства автоматизации. Такие методики, как непрерывное тестирование, непрерывная интеграция, комплексный мониторинг и возможности наблюдения за системой создают условия для раннего и автоматического обнаружения, обеспечивают прозрачность и получение быстрой обратной связи. Это позволяет исправлять ошибки раньше, чем при формальных проверках.

Непрерывная поставка предполагает более совершенный, по сравнению с традиционными, подход к управлению рисками, хотя совету по изменениям (САВ) в нем все еще отводится важная роль. По мере усложнения организаций крайне важно поддерживать информирование и координацию действий между командами. Наше предыдущее исследование показало, что развитие информационных потоков крайне важно для формирования высокоэффективной культуры ориентированной на конкретные задачи. Так как в рамках модели непрерывной поставки утверждать каждое отдельное изменение невозможно, здесь совет по изменениям должен сконцентрироваться на помощи командам по усовершенствованию процесса работы с целью повышения эффективности поставки программного обеспечения. Сюда относится предоставление рекомендаций и ресурсов для поддержки внедрения принципов, практик и подходов, стимулирующих эффективность. Совет по изменениям также может высказывать свою точку зрения по важным бизнес-решениям, требующим компромисса и утверждения руководством компании, например при выборе между скоростью вывода на рынок и снижением бизнес-рисков.

Как вы могли заметить, у совета по изменениям теперь новая стратегическая роль. Теперь, поручив подробные проверки кода специалистам и автоматизированным средствам, люди на руководящих постах могут потратить свое время и внимание на более стратегическую работу. Переход от вахтера к архитектору процесса и маяку в информационном пространстве — вот какой мы видим цель для развития внутренних подразделений управления изменениями. Это соответствует практикам, которых придерживаются организации с самыми высокими показателями эффективности поставки программного обеспечения.

## Прозрачный процесс реализации изменений

Несмотря на то, что конечной целью является отход от традиционных формальных процессов управления изменениями, даже простые улучшения коммуникации в текущем процессе и помочь командам сориентироваться в нем уже положительно влияют на эффективность поставки программного обеспечения. Когда у членов команды есть четкое понимание процесса согласования изменений для внедрения, повышается и эффективность и производительность. Это дает им уверенность в том, что процесс согласования изменений будет пройден в срок. Кроме того, они четко знают, какая последовательность шагов нужна для перевода заявки из статуса «отправлено» в статус «утверждено», — всегда, по всем типам изменений, с которыми они обычно имеют дело. Респонденты с прозрачным процессом управления изменениями в 1,8 раза чаще относились к «elite performers» командам.

Для крупных организаций управление изменениями — одно из самых больших ограничений. Для его устранения требуется

работа на нескольких уровнях. Команды могут реализовать непрерывную интеграцию, непрерывное тестирование и проверку кода коллегами, чтобы как можно быстрее обнаруживать ошибки в поставках, соблюдая при этом принципы разделения полномочий. И только практикующие технические специалисты имеют право создавать и автоматизировать придуманные нами решения для управления изменениями, делая их быстрыми, надежными, тиражируемыми и проверяемыми.

Руководителям каждого уровня следует отходить от формального процесса согласования, в котором внешние комитеты играют роль вахтеров для изменений, вместо этого переходя к управлению и развитию навыков. В конце концов, только руководители могут влиять на определенные уровни организационной политики и менять их. Мы зафиксировали существенное повышение эффективности — пропускной способности, стабильности и доступности — уже через несколько месяцев сотрудничества практикующих технических специалистов и руководителей организации.

# КУЛЬТУРА ПСИХОЛОГИЧЕСКОЙ БЕЗОПАСНОСТИ

Специалисты и лидеры, возглавляющие внедрение DevOps и технологическую трансформацию в организациях, часто называют культуру ключевым фактором успеха таких инициатив. Так, в своей книге «Философия DevOps. Искусство управления ИТ» авторы называют культуру основным условием успеха и масштабируемости технологических проектов.<sup>24</sup> Наше собственное исследование показало, что организационная культура, оптимизирующая информационные потоки, поощряющая доверие, инновации и разделение рисков, характерна для компаний с эффективной поставкой ПО и операционной деятельностью.<sup>25</sup>

Двухлетнее исследование в Google дало аналогичные результаты<sup>26</sup>: высокоэффективным командам нужна культура доверия и психологической безопасности, для них важна ясность и значимость выполняемой работы. Атмосфера команды позволяет ее участникам брать на себя просчитанные и умеренные риски, открыто выражать свое мнение и более творчески подходить к решению задач. Многим стало интересно, справедливы ли эти результаты вне корпорации Google. Даст ли такая культура преимущества другим компаниям с учетом того, с каким широким разнообразием компаний, инструментов и инженерных навыков мы имеем дело? Или это работает только для инженеров, прошедших невероятно строгие собеседования в Google, либо только для работающих в крупных предприятиях с огромной инфраструктурой и определенной кодовой базой?

24 Davis, J., & Daniels, R. (2016). Effective DevOps: building a culture of collaboration, affinity, and tooling at scale. O'Reilly Media, Inc.

25 В основе этого исследования — структура организационной культуры, изначально предложенная социологом Роном Веструмом (Dr. Ron Westrum). Эта модель описана в [отчете Accelerate State of DevOps 2018 года](#).

26 <https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/>

Чтобы ответить на этот вопрос, мы использовали для оценки культурных показателей вопросы, которые включала в свое исследование команда проекта «Аристотель».<sup>27</sup> Наш анализ показал, что культура психологической безопасности является показателем эффективности поставки программного обеспечения, эффективности организаций и производительности. Хотя в проекте «Аристотель» измерялись другие параметры, результаты свидетельствовали о том, что группы, работающие в атмосфере доверия и психологической безопасности, где есть ясность и видна значимость их работы, имеют преимущество и вне Google.



## ВЛИЯНИЕ ТЕХНОЛОГИЙ НА ЭФФЕКТИВНОСТЬ ОРГАНИЗАЦИЙ

Способность быстро и надежно поставлять программное обеспечение, гарантируя высокий уровень доступности — это веский аргумент. Такая возможность позволяет организациям легко и быстро прототипировать новые продукты и фичи, а также тестировать их влияние на пользователей, не затрагивая при этом реально существующих пользователей. Также она помогает организациям вовремя реагировать на изменения в требованиях регуляторов, быстро и надежно поставлять важные исправления и обновления программного обеспечения, необходимые для обеспечения безопасности. Если с умом использовать это преимущество, организации, показывающие высокую эффективность поставки ПО и операционной деятельности, смогут лучше реагировать на технологические и рыночные изменения, создавая превосходные товары и услуги. Это, в свою очередь, поможет организациям достигать желаемых результатов, как коммерческих, так и некоммерческих, оптимальными способами. Наш анализ этого года показал, что у «elite performers» команд в два раза больше шансов выполнить и превзойти цели по операционной эффективности.

Используемые нами показатели эффективности компаний составлены на основе научной литературы и охватывают как коммерческие<sup>28</sup>, так и некоммерческие<sup>29</sup> цели, в том числе:

- Рентабельность
- Производительность
- Доля рынка
- Количество клиентов
- Качество продуктов и услуг
- Операционная эффективность
- Удовлетворенность клиентов
- Качество предоставляемых продуктов и услуг
- Достижение организационных или заявленных в миссии целей

Как и в прошлом году, элемент второго порядка «Эффективности поставки ПО и операционной деятельности» позволяет прогнозировать эффективность компаний, причем лучше, чем только на основании показателей эффективности поставки программного обеспечения или его доступности.

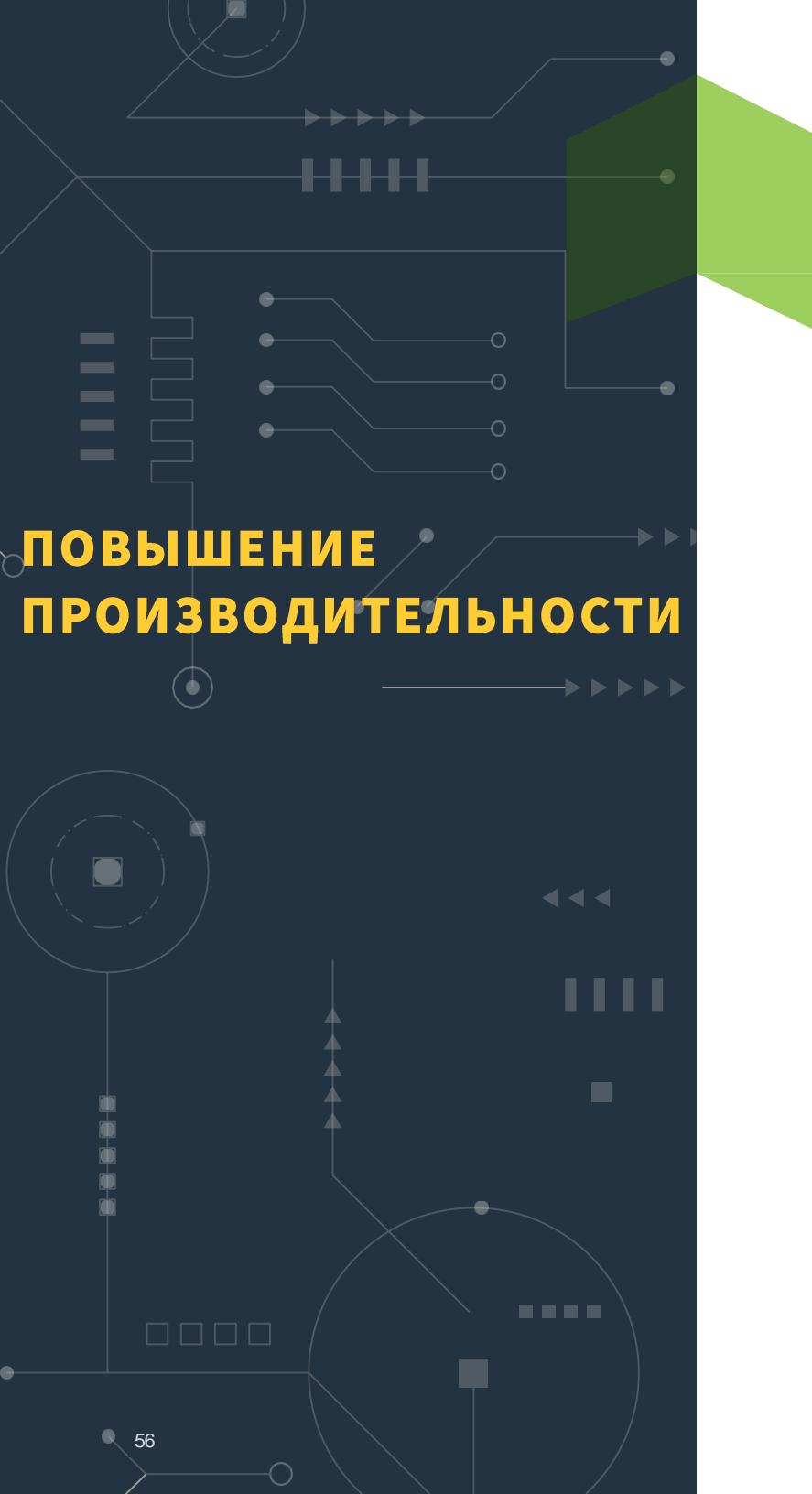
28 Widener, S. K. (2007). An empirical analysis of the levers of control framework. *Журнал Accounting, Organizations and Society*, 32(7-8), 757-788.

29 Cavalluzzo, K. S., & Ittner, C. D. (2004). Implementing performance measurement innovations: evidence from government. *Журнал Accounting, Organizations and Society*, 29(3-4), 243-267.

# КАК ПОВЫСИТЬ ПРОИЗВОДИТЕЛЬНОСТЬ?



Еще одна важная цель для команд и организаций — это повышение производительности для получения максимальной ценности от трансформации и от сотрудников компании. В этом году мы впервые исследуем производительность: каким образом организации поддерживают ее, разумно инвестируя в инструменты и информацию, как ей мешает технический долг и как она влияет на баланс между работой и личной жизни сотрудников и на эмоциональное выгорание.



## ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ

Большинство опрошенных признает безусловный приоритет высокой производительности. Продуктивно работающие инженеры выполняют свои задачи более эффективно и получают возможность реинвестировать оставшееся время в другие проекты<sup>30</sup>, такие как создание документации и рефакторинг, или просто успевают выполнять больше своих непосредственных задач для поставки дополнительных фич или постройки дополнительной инфраструктуры.<sup>31</sup>

Но что такое производительность и как ее измерить? Производительность нельзя описать простым показателем, таким как количество строк кода, количество стори-поинтов или закрытых багов; иначе мы получим нежелательные последствия, которые вредят общим целям команды.<sup>32</sup> Например, команда может отказать в просьбе о помощи, так как это отрицательно скажется на ее скорости, даже если эта помощь важна для достижения организационных целей.

---

30 Рекомендуем ознакомиться с идеями о производительности, высказанными известным экономистом Хэлом Варианом (Hal Varian). <https://www.wsj.com/articles/silicon-valley-doesnt-believe-u-s-productivity-is-down-1437100700>

31 Мы обсуждаем преимущества реинвестирования времени, сэкономленного в результате повышения производительности (а не считаем эту ситуацию экономией затрат, которые в любом случае нужно сокращать) в публикации по рентабельности «Прогнозирование ценности от внедрения DevOps» (Forecasting the Value of DevOps Transformations) 2017 года. Эта идея не нова. Экономисты уже исследовали этот вопрос, а Хэл Вариан касается его в указанной выше статье в WSJ. [cloud.google.com/devops](http://cloud.google.com/devops)

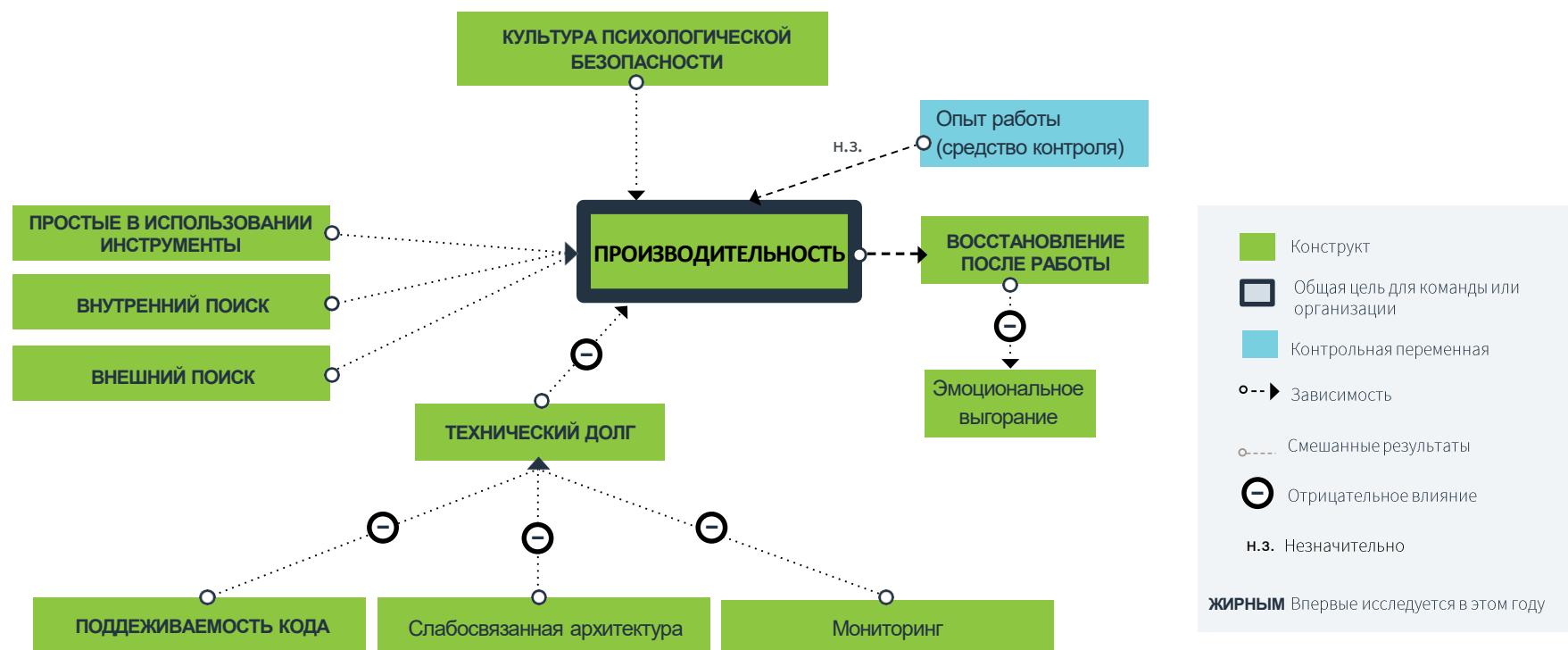
32 Закон Гудхарта: «Когда показатель становится целью, он перестает быть хорошим показателем». Концентрируясь на простых в измерении локальных показателях, команды часто выбирают их в качестве своих целей, жертвуя глобальными целями, важными для обеспечения организационных результатов. Более подробные сведения см. в главе 2 книги *Accelerate: The Science of Lean Software and DevOps*

Исследователи давно обсуждают этот вопрос, и большинство из них пришло к тому же выводу.

## Высокая производительность — это способность выполнять сложные и трудоемкие задачи минимально отвлекаясь и прерываясь.

Многие из нас описали бы этот процесс как «нахождение в состоянии потока» или «вхождение в ритм».

Чтобы использовать эту модель, найдите на графике цель, по которой вы хотите достичь улучшений, а затем определите влияющие на нее факторы. Например, если ваша цель — сокращение технического долга, на нее будет влиять поддерживаемость кода, наличие слабосвязанной архитектуры и мониторинг.



# ПОЛЕЗНЫЕ И ПРОСТЫЕ В ИСПОЛЬЗОВАНИИ ИНСТРУМЕНТЫ

Полезные и простые в использовании инструменты — необходимость в технологических сферах, предназначенных для конечного потребителя. Однако эти очевидные характеристики часто остаются вне поля зрения технических экспертов, предполагающих, что они отлично во всем разбираются и могут успешно работать с любым инструментом или технологией (или потому что лица, ответственные за закупку инструментов для этих групп, считают удобство использования не столь значимым для технических специалистов либо делают свой выбор по таким параметрам, как стоимость, условия лицензирования или система управления поставщиками). В действительности справедливо обратное: когда речь идет про построение сложных систем и управление критичной для бизнеса инфраструктурой инструменты приобретают еще большую важность из-за сложности самой работы.

Мы сосредоточили внимание на инструментах, используемых при развертывании программного обеспечения через процессы непрерывной интеграции/поставки и автоматизации тестирования, потому что они лежат в основе DevOps. Мы обнаружили, что на производительность влияют следующие два атрибута:

- Насколько легко использовать инструменты (включая четкость и простоту взаимодействия и эксплуатации);
- Насколько полезны инструменты для выполнения целей, связанных с этой работой.

## НАИБОЛЕЕ ПРОИЗВОДИТЕЛЬНЫЕ ИНЖЕНЕРЫ



**1.5**  
РАЗ ЧАЩЕ

пользуются простыми в использовании инструментами

# ИСПОЛЬЗОВАНИЕ ИНСТРУМЕНТОВ ПО ПРОФИЛЯМ ЭФФЕКТИВНОСТИ

	Low	Medium	High	Elite
Комбинация проприетарных инструментов, программного обеспечения с открытым исходным кодом и коробочных решений	<b>30%</b>	<b>34%</b>	<b>32%</b>	<b>33%</b>
Преимущественно программное обеспечение с открытым исходным кодом и коробочные решения со значительными доработками	<b>17%</b>	<b>8%</b>	<b>7%</b>	<b>10%</b>
Преимущественно программное обеспечение с открытым исходным кодом и коробочные решения с незначительными доработками	<b>14%</b>	<b>21%</b>	<b>18%</b>	<b>20%</b>
В основном коробочные решения	<b>8%</b>	<b>12%</b>	<b>8%</b>	<b>4%</b>
В основном программное обеспечение собственной разработки, используемое только в моей организации	<b>20%</b>	<b>6%</b>	<b>5%</b>	<b>6%</b>
Преимущественно программное обеспечение с открытым исходным кодом со значительными доработками	<b>6%</b>	<b>7%</b>	<b>5%</b>	<b>12%</b>
Преимущественно программное обеспечение с открытым исходным кодом с незначительными доработками	<b>5%</b>	<b>12%</b>	<b>24%</b>	<b>15%</b>

А как выглядят эти инструменты? Мы подробно изучили данные по профилям эффективности и заметили ряд любопытных закономерностей.

- Самая высокая концентрация проприетарного программного обеспечения наблюдается у «low performers», в то время как самая низкая — у «high performers» и «elite performers». Проприетарное ПО может быть полезным, однако его техническое обслуживание и поддержка обходятся довольно дорого. Неудивительно, что наиболее эффективные компании уже отошли от этой модели.
- Концентрация коробочных продуктов с небольшими доработками примерно одинакова у всех. Некоторые могут удивляться почему же «high performers» могут использовать коробочные решения и сохранять высокую эффективность, особенно на фоне представления о том, что «программное обеспечение правит балом» и мы должны создавать свои собственные приложения?

# АВТОМАТИЗАЦИЯ И ИНТЕГРАЦИЯ ПО ПРОФИЛЯМ ЭФФЕКТИВНОСТИ

	Low	Medium	High	Elite
Автоматизированная сборка	<b>64%</b>	<b>81%</b>	<b>91%</b>	<b>92%</b>
Автоматизированное модульное тестирование	<b>57%</b>	<b>66%</b>	<b>84%</b>	<b>87%</b>
Автоматизированное приемочное тестирование	<b>28%</b>	<b>38%</b>	<b>48%</b>	<b>58%</b>
Автоматизированное тестирование производительности	<b>18%</b>	<b>23%</b>	<b>18%</b>	<b>28%</b>
Автоматизированное тестирование безопасности	<b>15%</b>	<b>28%</b>	<b>25%</b>	<b>31%</b>
Автоматизированное выделение ресурсов и развертывание в тестовых средах	<b>39%</b>	<b>54%</b>	<b>68%</b>	<b>72%</b>
Автоматизированное развертывание в промышленной среде	<b>17%</b>	<b>38%</b>	<b>60%</b>	<b>69%</b>
Интеграция с чатботами/Slack	<b>29%</b>	<b>33%</b>	<b>24%</b>	<b>69%</b>
Интеграция с инструментами мониторинга и наблюдения за промышленными средами	<b>13%</b>	<b>23%</b>	<b>41%</b>	<b>57%</b>
Ничего из вышеуказанного	<b>9%</b>	<b>14%</b>	<b>5%</b>	<b>4%</b>

33 Мартин Фоулер, MartinFowler.com, «Дихотомия стандартного и стратегического» (Utility Vs Strategic Dichotomy). <https://martinfowler.com/bikiki/UtilityVsStrategicDichotomy.html>

34 Передовая практика для инженера по надежности (SRE): сокращайте объем утомительной работы, т. е. работы, выполняемой с низкой производительностью.

Как отмечает Мартин Фоулер<sup>33</sup>, компаниям следует учитывать, какое программное обеспечение является стратегическим инструментом, а какое — просто полезным. Удовлетворяя стандартные потребности за счет коробочных решений с минимальными доработками, эффективные компании экономят ресурсы, сохраняя их для разработки стратегического важного ПО.

Мы также заметили, что «elite performers» команды гораздо чаще используют автоматизацию и интегрируют инструменты в цепочки инструментов практически по всем направлениям. Хотя многие считают автоматизацию слишком дорогой в реализации (мы часто слышим нечто вроде «у меня нет ни времени, ни бюджета на автоматизацию, ведь это не фича для наших клиентов»), это весьма разумная инвестиция.<sup>34</sup> Она позволяет инженерам тратить меньше времени на выполнение ручной работы, тем самым высвобождая его для других важных задач, таких как разработка нового функционала, рефакторинг, проектирование и создание документации. Также она дает инженерам уверенность в надежности набора инструментов, снижая уровень стресса при отправке изменений.

## Технические эксперты и инструменты

В рамках исследования 2017 года мы обнаружили, что команды, уполномоченные принимать самостоятельные решения о выборе инструментов и внедрениях, вносят ощутимый вклад в повышение эффективности поставки программного обеспечения. Исследование этого года показало, что при наличии такой возможности «high performers» выбирают полезные и удобные в использовании инструменты, которые повышают производительность.

Это имеет важные последствия для разработки продукта. Продукты, сочетающие пользу и удобство использования, с большей вероятностью будут применяться техническими экспертами, а при использовании — обеспечивать более высокие результаты. Лидерам отрасли следует отдавать приоритет именно таким видам инструментов.

Недостаточно поставлять продукты, у которых есть все необходимые функции; чтобы продукт применяли и он мог приносить пользу в процессе DevOps-трансформации, он должен быть удобным в использовании.

## ПРОИЗВОДИТЕЛЬНОСТЬ, ЭМОЦИОНАЛЬНОЕ ВЫГОРАНИЕ И МНОГОЗАДАЧНОСТЬ

Нас интересовало, насколько ли отличаются количество одновременных задач в работе у самых эффективных и отстающих. В конце концов, производительность — это способность выполнять работу до конца и ощущать себя в состоянии потока.

Чтобы выяснить это, мы задали респондентам несколько вопросов.

- Сколько ролей они совмещают или сколько видов работ они выполняют независимо от официальной должности
- Между сколькими проектами им приходится переключаться в течение дня
- Над сколькими проектами в целом они работают

К нашему удивлению, больших различий между «low», «medium», «high» и «elite performers» мы не увидели.

Следовательно, мы не можем утверждать, что эффективность разработки и поставки программного обеспечения влияет на количество ролей и проектов, между которыми переключаются респонденты. Здесь нельзя сказать: «Преодолейте эту фазу, и все будет значительно лучше». Вместо этого нам нужно двигаться к обеспечению устойчивости результатов своей работы. Это делается через совершенствование процессов и автоматизацию, которые сокращают объем тяжелой и утомительной работы, делая задачи повторяемыми, согласованными, быстро выполнимыми, масштабируемыми и проверяемыми. Также за счет этого у нас высвобождается время на новые, более творческие задачи.

# ВНУТРЕННИЙ И ВНЕШНИЙ ПОИСК

Поиск нужной информации, которая поможет решить проблему, устранить ошибку или легко и быстро найти решение аналогичного вопроса, может стать ключевым фактором при выполнении задач и поддержке потока работы. Это особенно справедливо в современной технологической среде, состоящей из сложных систем. Мы обнаружили, что наличие доступа к информации поддерживает производительность. Источники информации делятся на две категории: внутренние и внешние.

- **Внутренний поиск.** Инвестиции в разработку документации и создание кода, а также в эффективный поиск по корпоративным базам знаний, репозиториям, системам обработки заявок и документам помогают повысить инженерную производительность. Те, кто использует поиск по внутренним источникам информации, демонстрируют высокую производительность в 1,73 раза чаще. Предоставление разработчикам, системным администраторам и персоналу службы поддержки возможности поиска по внутренним ресурсам помогает им находить ответы в контексте своей работы (например, с помощью функции «найти похожее») и быстрее применять решения. Кроме того, внутренние базы знаний, которые должным образом поддерживаются и ведутся, создают возможности для дополнительного обмена информацией и фиксации знаний.

## ИСПОЛЬЗУЮЩИЕ ВНУТРЕННИЙ ПОИСК



в **1.73**  
раза

более производительные

## ИСПОЛЬЗУЮЩИЕ ВНЕШНИЙ ПОИСК

в **1.67**  
раза



более производительные

Например, найдя решение, которое почти подходит, сотрудники могут задать уточняющий вопрос или инициировать обсуждение во внутреннем сообществе, в результате чего база знаний пополняется новой информацией. Если организация инвестировала в системы с удобным поиском по всем типам информации и данных, соответствующая культура может поддержать «добродетельный цикл» обмена знаниями. Некоторые организации используют во внутреннем поиске технологии машинного обучения для выявления и рекомендации возможных решений.

- Внешний поиск. Сюда относятся внешние источники данных, такие как поисковики или Stack Overflow. Наш анализ показал, что сотрудники, использующие внешний поиск в своей работе, в 1,67 раза чаще чувствуют, что они более продуктивны. Внешний поиск важен как основа сообществ для обучения и развития (и имеет важное побочное преимущество в виде удобства найма), а также поддерживает использование и применение публичного облака и

инструментов с открытым исходным кодом. Т. е. используя распространенные внешние инструменты и системы с сильным сообществом пользователей и надежной экосистемой, в процессе поиска и устранения неисправностей технические эксперты могут получать помощь от специалистов со всего мира; в то время как по закрытым решениям или системам собственной разработки им доступно лишь мнение внутренних специалистов внутри организации. Мы находим подтверждения этому в данных. Согласно исследованию 2018 года, «elite performers» команды использовали решения с открытым исходным кодом и планировали расширять сферу их применения. В исследовании этого года мы видим, что «elite performers» использует больше инструментов с открытым исходным кодом, а «low performers» — активнее всего используют проприетарное программное обеспечение (см. [страницу 59](#)); подобный выбор технологий оказывает влияние на производительность.

# ТЕХНИЧЕСКИЙ ДОЛГ

Понятие технический долг ввел в 1992 году Уорд Каннингем (Ward Cunningham)<sup>35</sup>, чтобы описать, что происходит, когда не удается должным образом поддерживать то, что он называет «незрелым» кодом.

*Хотя незрелый код может отлично работать и быть абсолютно приемлемым с точки зрения клиента, его чрезмерное количество затрудняет освоение программы, из-за чего для ее обслуживания требуются программисты со слишком узкой специализацией, а сам продукт теряет гибкость. Поставлять сырой код — это как влезть в долги. Небольшой долг ускоряет разработку, пока его вовремя возвращают путем переписывания кода... Опасность возникает, когда долг не возвращается. Каждая минута, потраченная на не совсем правильный код, считается процентами по этому долгу. Под долговой нагрузкой неконсолидированных решений может встать работа целых отделов проектирования.*

В современных сложных системах технический долг может возникать в скриптах, файлах конфигурации и инфраструктуре, а также в коде приложений. Техническим долгом считаются код или системы, для которых характерны:

- Известные ошибки, исправлением которых пренебрегли в угоду новым функциям;
- Недостаточное покрытие тестами;
- Проблемы, связанные с низким качеством кода или неудачной архитектурой;
- Код или артефакты, не удаленные после долгого неиспользования;
- Внедрения, по которым у действующей команды нет знаний и опыта, и соответственно, которые она не сможет эффективно отлаживать или обслуживать;
- Незавершенная миграция;
- Устаревшие технологии;
- Неполная или устаревшая документация или отсутствие комментариев.

<sup>35</sup> <http://c2.com/doc/oopsla92.html>



## АКТИВНОЕ СОКРАЩЕНИЕ ТЕХНИЧЕСКОГО ДОЛГА

Мы обнаружили, что технический долг отрицательно влияет на производительность. Респонденты с большим объемом технического долга имели в 1,6 раза меньшую производительность, а самые эффективные организации в 1,4 раза чаще имели небольшой технический долг.

### Работа с техническим долгом

Уорд Каннингем заявляет, что «лучшим антидотом [для изменения системы] является более полное знакомство с продуктом и его реализацией».

Когда инженеры понимают изменения, они могут их принять. Специфика современных сложных инфраструктур и распределенных систем не позволяет инженерам держать в голове полную модель системы. Кроме того, для поддержки этих сложных систем требуются эксперты с более узкой специализацией, которые не могут знать систему в целом.<sup>36</sup>

Мы можем помочь инженерам получить представление о системе, проектируя гибкие, расширяемые и прозрачные системы с целью сокращения технического долга.

Как же на самом деле сократить технический долг, а не просто работать с ним? Один из подходов — рефакторинг. Рефакторинг — это «дисциплинированный подход к реорганизации существующего кода, изменению внутренней структуры без изменения внешнего поведения». Мартин Фаулер отмечает, что рефакторинг должен стать частью повседневной работы. Также немаловажно наличие более совершенных инструментов со встроенными надежными средствами поддержки рефакторинга. Фактически, многие крупные организации инвестировали в инструменты для реорганизации кода в своей базе; например, Facebook открыл исходный код своего инструмента [fastmod](#), а Google — [ClangMR](#).

<sup>36</sup> Чтобы иметь полное представление о работе системы, нужно быть full-stack-разработчиком. Помимо обычных сложностей с терминологией, что такое полный стек (full stack)? От чипа до CSS? Многие в отрасли придерживаются мнения, что стать full-stack-разработчиком невозможно или нежелательно..

## КУЛЬТУРА ПСИХОЛОГИЧЕСКОЙ БЕЗОПАСНОСТИ

Культура, в которой ценится психологическая безопасность, доверие и уважение, поддерживает производительность, позволяя сотрудникам сосредоточиться на решении рабочих задач вместо соперничества и участия в политических играх. Эти выводы перекликаются с работами других исследователей; как обсуждалось в предыдущем разделе, исследование Google показало, что подобная культура помогает командам быть более эффективными.

## ПРОИЗВОДИТЕЛЬНОСТЬ И ПРОЕКТЫ С ОТКРЫтыМ ИСХОДНЫМ КОДОМ

Эти выводы по производительности также применимы к проектам с открытым исходным кодом. Производительность участников таких проектов быстрее увеличивается от нулевой до хорошей, если инструкции по контрибуции являются актуальными, простыми и соответствуют используемым в других проектах с открытым исходным кодом. К крупному проекту с уникальным и устаревшим процессом контрибуции будет непросто присоединяться новичкам из-за высокого порога входления. Прибавьте к этому технический долг, и участники будут рассматривать ваш проект в режиме «только чтение». Применение всех указанных выше передовых практик в проектах с открытым исходным кодом поможет новым участникам быстрее войти в курс дела и более эффективно развивать сообщество.

# ДОПОЛНИТЕЛЬНЫЕ ПРЕИМУЩЕСТВА ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ

Преимущества для команды и организации от более высокой производительности, как правило, очевидны: ей удается сделать больше работы и создать больше ценности. Но какие преимущества получают люди, выполняющие эту работу?

## Восстановление после работы

Исследование показало, что высокая производительность положительно влияет на восстановление после работы, что подтверждают и наши данные. Восстановление после работы — это способность справляться с рабочим стрессом и абстрагироваться от работы, когда вы не работаете. Это еще называют умением «оставлять работу на работе». Исследование показало, что сотрудники, у которых получается не думать о работе дома, лучше себя чувствуют и лучше справляются с рабочим стрессом. Важно и обратное: если сотрудник чувствует переутомление, ему труднее не думать о работе, что приводит к эмоциональному выгоранию и снижению удовлетворенности жизнью.<sup>37</sup>

## Эмоциональное выгорание

Всемирная организация здравоохранения признает эмоциональным выгоранием состояние, приводящее к неконтролируемому хроническому стрессу на рабочем месте<sup>38</sup>; и это нечто большее, чем просто усталость. Эмоциональное выгорание — это сочетание истощения, циничного отношения и неэффективности на работе. Оно характерно для сложной работы, связанной с высоким уровнем риска, например в больницах, авиадиспетчерских службах и

37 Sonnentag, S., & Fritz, C. (2015). Recovery from job stress: The stressor-detachment model as an integrative framework. *Journal of Organizational Behavior*, 36(S1), S72-S103.  
<https://onlinelibrary.wiley.com/doi/abs/10.1002/job.1924>

38 Всемирная организация здравоохранения, Международная классификация болезней: «профессиональный синдром» эмоционального выгорания.  
[https://www.who.int/mental\\_health/evidence/burn-out/en/](https://www.who.int/mental_health/evidence/burn-out/en/)

в сфере высоких технологий.<sup>39</sup> Исследование показало, что связанная со стрессом работа вредит физическому здоровью так же, как пассивное курение<sup>40</sup> и ожирение.<sup>41</sup> В особенно тяжелых случаях эмоциональное выгорание приводит к проблемам в семье, клинической депрессии и даже суициду.

В отчете этого года мы обнаружили, что способность восстанавливаться после работы может снизить вероятность эмоционального выгорания, что подтвердили и другие исследования. Также мы проверили результаты исследований прошлых лет и выяснили, что эффективные технические практики и более совершенные процессы (в виде понятного управления изменениями) могут снижать вероятность эмоционального выгорания. По полученным данным, самые эффективные команды в два раза реже испытывают ощущение выгорания. Иными словами, «low performers» в два раза чаще испытывают ощущение выгорания.

39 Хотя некоторые заявляют, что сфера высоких технологий не связана с высоким риском, стресс от публикации кода с ошибками вполне реален. От ошибок, которые могут привести к широкой огласке или иметь роковые последствия (например, при создании программного обеспечения для больниц и систем здравоохранения), не застрахованы многие из тех, кто занят в индустрии высоких технологий.

40 Goh, J., Pfeffer, J., Zenios, S. A., & Rajpal, S. (2015). Workplace stressors & health outcomes: Health policy for the workplace. Журнал *Behavioral Science & Policy*, 1(1), 43-52.

41 Chandola, T., Brunner, E., & Marmot, M. (2006). Chronic stress at work and the metabolic syndrome: prospective study. Журнал *BMJ*, 332(7540), 521-525.

## КАК ПОДДЕРЖАТЬ СПОСОБНОСТЬ ВОССТАНАВЛИВАТЬСЯ ПОСЛЕ РАБОТЫ

Высокая производительность положительно влияет на способность восстанавливаться после работы. Так как это один из ключевых факторов снижения стресса и риска эмоционального выгорания, каким образом его можно обеспечить?

Исследование показало, что существует пять способов улучшить восстановление после работы:

**Психологическое переключение** — это способность не думать о работе вне работы. Рассмотрите возможность электронных барьеров, которые помогут удержать рабочий стресс на работе.

**Расслабление** — это не только роскошь, которую мы можем позволить себе во время отпуска, а ключевой компонент производительности. Сделайте привычкой отводить время на восстановление.

**Развитие навыков**, не связанных с работой, поддерживает положительное отношение к происходящему, снижает уровень стресса и помогает поддерживать здоровые отношения. Практикуйте не профессиональные навыки, которые доставляют вам удовольствие.

**Нехватка контроля** вызывает стресс. Балансируйте рабочие задачи, которые трудно контролировать, с активными занятиями вне работы, которые всецело находятся в вашей власти.

Поощряйте привычку на время отключаться от работы. Руководство, в частности, может задать режим работы, предполагающий, что сотрудники вовремя уходят домой, не работают вечерами и по выходным. Специалисты могут придерживаться принципа отключения от работы, находясь вне офиса, и призывать коллег поступать так же.\*

\* Культура психологической безопасности тесно связана с восстановлением после работы и может помочь в поддержке последней. Это правда, потому что члены команды могут спокойно говорить о стрессе и нагрузке, а также делать перерывы в работе, когда уходят домой. Мы не проверяли зависимости, потому что используемые нами показатели, связанные с культурой, не содержат вопросов о «поддержке или поощрении перерывов в работе» и иных моделях поведения, которые укрепляют способность восстанавливаться после работы.

# МЕТОДИКИ ТРАНСФОРМАЦИИ: ЧТО РЕАЛЬНО РАБОТАЕТ



Нас часто спрашивают, каким образом новые методы работы распространяются внутри организаций. Непрерывные реструктуризации и реорганизации не могут обеспечить устойчивость, поскольку в краткосрочной перспективе отрицательно сказываются на производительности. Нам известно о нескольких компаниях, осуществивших масштабную трансформацию без реорганизации. И хотя единственно правильного пути к успеху не существует, кое-что все же очевидно: трансформация DevOps — не пассивное явление. В этом году мы постарались определить наиболее популярные практики в области распространения передовых принципов DevOps в организациях.

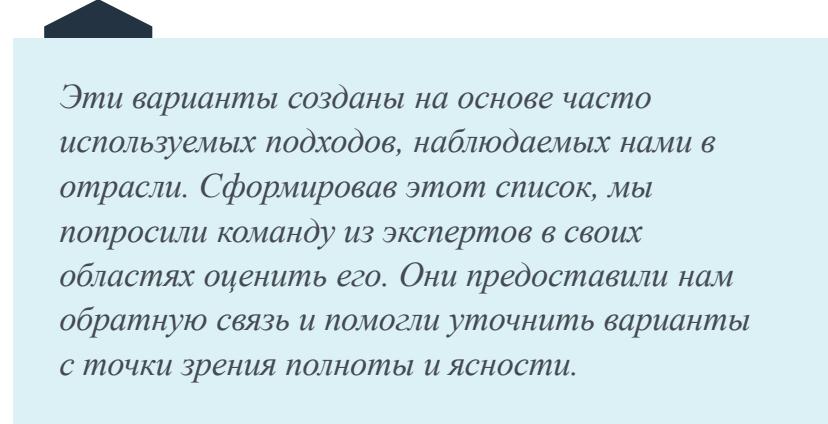


## ТРАНСФОРМАЦИЯ: ЧТО РЕАЛЬНО РАБОТАЕТ

Мы попросили респондентов указать, как в их командах и организациях распространяются подходы DevOps и Agile.

Респонденты могли выбрать один или несколько из следующих подходов \*:

- Учебный центр (иногда называется «додзё»)
- Центры компетенций
- Проверка концепции, не имеющая продолжения
- Проверка концепции как шаблон
- Проверка концепции как отправная точка
- Сообщества по практикам
- «Большой взрыв»
- Подход «снизу вверх», или «Инициатива снизу»
- Смешанный подход



*Эти варианты созданы на основе часто используемых подходов, наблюдаемых нами в отрасли. Сформировав этот список, мы попросили команду из экспертов в своих областях оценить его. Они предоставили нам обратную связь и помогли уточнить варианты с точки зрения полноты и ясности.*

\* В Приложении В содержится подробное описание каждого подхода.



Мы исследовали данные согласно группам эффективности, чтобы проанализировать эффективность каждой стратегии. Это не идеальный расчет, но он дает представление о том, что делают наиболее и наименее эффективные команды для масштабирования своей технологической трансформации.

Смешанный подход обычно занимает в этой выборке 40%, при этом финансирования и ресурсов не хватает, чтобы сосредоточить усилия на какой-то конкретной области. Мы предупреждаем, что без стратегии, определяющей курс цифровой трансформации, организации будут часто ошибаться, пытаясь подстраховаться и страдать от переизбытка инициатив в различных направлениях. Это неизбежно ведет к недоснабжению ресурсами важных работ и обрекает их на провал. Рекомендуется выбрать ограниченное число инициатив и, чтобы довести их до конца, выделить достаточно ресурсов (т. е. времени, денег, поддержки руководства и ведущих специалистов).<sup>42</sup>

<sup>42</sup> Это во многом напоминает описанную ранее модель ограничений в концепции непрерывных улучшений. Ставьте краткосрочные и долгосрочные цели, определяйте основные области, в которых для достижения этих целей требуется больше всего улучшений, и выделяйте ресурсы соответствующим образом.

«High performers» отдают предпочтение стратегиям, формирующими сообщества на нижнем и верхнем уровнях организации, которые, по всей видимости, делают их более устойчивыми к реорганизациям и изменениям в продукте. Две основные применяемые стратегии: сообщества по практикам и «инициатива снизу», за ними следуют проверка концепции как (тиражируемый) шаблон и проверка концепции как отправная точка.

«Low performers» тяготеют к учебным центрам (также известным как «додзё») и центрам компетенций — стратегиям, создающим более изолированные знания и опыт. Также они пробуют использовать проверку концепции которая, как правило, не имеют продолжения и не приводят к успеху.

У некоторых стратегий есть общие черты во всех профилях эффективности: Во всех группах применяют и развивают смесь различных стратегий.



Ни для одной из групп не характерно активное использование стратегии «Большой взрыв» — чаще остальных ее используют «low performers» (19% времени) — что, скорее всего, к лучшему. По нашему опыту, это очень трудная модель с точки зрения реализации. К ней стоит прибегать только в самых тяжелых случаях, когда требуется «полная перезагрузка». Стратегия «Большого взрыва» предполагает, что каждый должен включиться в процесс надолго, а ресурсы на нее выделяются как на многолетнюю инициативу. Это объясняет, почему данный метод чаще всего встречается у «отстающих».

## **Почему не рекомендуется использовать центры компетенций или учебные центры?**

В общем случае центры компетенций создавать не рекомендуется, поскольку они сосредоточивают опыт в одной группе. Это создает ряд проблем. Во-первых, центр компетенций становится узким местом, ограничивающим доступ организации к релевантным специальным знаниям. Он не может масштабироваться с ростом спроса на

соответствующие экспертные знания и навыки в организации. Во-вторых, он формирует эксклюзивную группу «экспертов» в организации, а не инклузивную группу коллег, которые продолжают совместно учиться и развиваться. Такая эксклюзивность способствует появлению нездоровых норм и моделей поведения и может подрывать основы эффективных организационных культур. И наконец, эксперты выпадают из своей основной работы.

Некоторые считают учебные центры успешным форматом, но для них требуются выделенные ресурсы и программы обучения, которые будут использоваться как для непосредственно обучения, так и для закрепления результатов обучения. Чтобы сделать свои программы обучения эффективными, многие компании выделяют отличных специалистов. Для создания отдельной творческой среды отводятся целые здания, а персонал занимается созданием учебных материалов и оценкой прогресса.



Для поддержки и распространения обучения в рамках организации требуются дополнительные ресурсы. Организация должна позаботиться о том, чтобы команды, посещающие Учебный центр, смогли применить на практике освоенные навыки и модели поведения в обычной рабочей среде, не возвращаясь к старым привычкам. Если такие ресурсы отсутствуют, велик риск того, что соответствующие инвестиции были сделаны организацией напрасно.

Предполагается, что команды посещают Центр для изучения новых технологий и процессов и соответствующие знания будут распространены по всей организации. На деле новые навыки остаются в Центре, оставаясь изолированными, пусть даже и непродолжительное время. Аналогичные ограничения свойственны центрам компетенций. Какие последствия ждут нас, если персонал учебного центра (или другие оторванные от жизни «эксперты») занят лишь разработкой семинаров и учебных материалов и не выполняет соответствующие задачи на практике?

# ТЕПЛОВАЯ КАРТА СТРАТЕГИЙ DEVOPS-ТРАНСФОРМАЦИИ ПО ПРОФИЛЯМ ЭФФЕКТИВНОСТИ

	Low	Medium	High	Elite
Учебный центр	27%	21%	18%	14%
Центр компетенций	34%	34%	20%	24%
Проверка концепции, не имеющая продолжения	41%	32%	20%	16%
Проверка концепции как шаблон	16%	29%	29%	30%
Проверка концепции как отправная точка	21%	24%	29%	30%
Сообщества по практикам	24%	51%	47%	57%
«Большой взрыв»	19%	19%	11%	9%
Подход «снизу вверх», или «Инициатива снизу»	29%	39%	46%	46%
Смешанный подход	46%	42%	34%	38%

## Масштабирование работающих стратегий

Мы провели дополнительный кластерный анализ, чтобы разобраться в стратегиях, наиболее часто используемых «high» и «elite performers» командами, и выделить четыре закономерные модели.

- **Строители сообществ.** Эта группа уделяет основное внимание сообществам по практикам, «инициативам снизу» и проверкам концепций (как шаблон и как отправная точка, описанным ранее). Это происходит в 46 % случаев.
- **Университет.** Эта группа уделяет основное внимание обучению и тренингам. Большая часть подобных инициатив сосредоточена в центрах компетенций, сообществах по практикам и учебных центрах. Мы встречаем такую модель только в 9% случаев. Т. е. эта стратегия может быть успешной, но не распространена, а также требует значительных инвестиций и планирования, чтобы результаты обучения распространялись по организации.
- **Новаторы.** Эта группа уделяет основное внимание «инициативам снизу» и

- сообществам по практикам. Эта группа оказалась самой автономной (hands-off) и встречается в 23% случаев.
- **Экспериментаторы.** Эта группа встречается в 22% случаев. Для нее характерна высокая активность по всем стратегиям, кроме «Большого взрыва» и «додзё», т. е. по всем мероприятиям, направленным на формирование сообществ. Также они используют подход проверки концепции, не имеющей продолжения. Их способность использовать преимущества этого подхода и сохранять при этом высокую эффективность говорит о том, что здесь эта стратегия используется для проведения экспериментов и быстрого тестирования новых идей.

Принимая во внимание эти четыре модели, мы можем начать разрабатывать стратегию DevOps-трансформации. «High» и «elite performers» начали разрабатывать стратегии для масштабирования, из которых организации могут перенять самые подходящие и повысить свою собственную эффективность.

# ЗАКЛЮЧЕНИЕ



Каждые десять лет в моду входит новая методология разработки программного обеспечения. Хотя каждая из них должна стать лучше предыдущих, история доказывает их неэффективность. Тем не менее, мы продолжаем получать подтверждения, что концепция DevOps дает преимущества. В течение шести лет подряд мы статистически проверяли основные практики и подходы, которые помогают организациям совершенствовать свои процессы разработки и поставки программного обеспечения с помощью методик DevOps.

DevOps — это не тренд. Эта методология постепенно станет стандартом разработки и эксплуатации программного обеспечения, предлагая улучшения каждому участнику этих процессов.

Мы благодарим всех, кто принял участие в опросе этого года, и надеемся, что наше исследование поможет вам и вашей организации формировать команды и разрабатывать ПО лучше, оставляя при этом работу на работе.



# МЕТОДОЛОГИЯ

Наши точные методы анализа не позволяют ограничиться сырьими цифрами — в отчете рассматриваются зависимости между эффективностью поставки ПО, эффективностью компаний, техническими практиками, культурными нормами и производительностью. В этом разделе мы описываем методы, на которых строится наш анализ, а также принципы отбора респондентов и разработки вопросов, моделей и элементов. Более подробную информацию см. в части II нашей книги [Accelerate: The Science of Lean Software and DevOps](#).

Вопросы о методологии опроса направляйте на адрес [dora-data@google.com](mailto:dora-data@google.com).

## Архитектура исследования

В данном исследовании используется теоретически обоснованный метод поперечных срезов. Этот теоретически обоснованный подход также известен как статистический вывод или индуктивная статистика. Это один из самых распространенных методов исследования в сфере бизнеса и технологий. Статистический вывод используется, когда невозможно провести эксперимент в чистом виде и предпочтительны полевые эксперименты, например при сборе данных в организациях со сложной структурой, а не в стерильных лабораторных условиях, когда компании не готовы жертвовать прибылью, чтобы подстроиться под контрольные группы, определенные исследовательской командой.

## Целевая аудитория и метод выборки

Целевая аудитория данного опроса — специалисты и руководители, работа которых тесно связана с технологиями и трансформацией, особенно те, кому знакома методология DevOps. У нас не было основного списка таких людей — мы могли описать

их, но не знали, где именно они работают, как их найти и сколько их, — поэтому для поиска респондентов мы использовали выборку по методу снежного кома. Это значит, что мы распространяли опрос с помощью рассылок по электронной почте, онлайн-кампаний и социальных сетей. Также мы просили людей делиться ссылкой на опрос с коллегами и знакомыми, увеличивая выборку по аналогии со снежным комом, который, скатываясь с горы, вбирает в себя дополнительный снег. Скорее всего, наша выборка ограничена организациями и рабочими группами, которые знакомы с DevOps и применяют эту методологию. Чтобы преодолеть ограничения выборки по методу снежного кома, важно было изначально иметь разнотипную выборку. Мы справились с этой задачей, используя для создания исходной выборки собственные и предоставленные спонсорами исследования списки контактов. В результате выборки демография и статистика по организациям побольшей части соответствовали отраслевым тенденциям.

## Создание неявных конструктов

Мы сформулировали наши гипотезы и конструкты, по возможности используя существующие проверенные конструкты. При создании новых конструктов мы руководствовались теорией, определениями и мнениями экспертов. Затем мы приложили дополнительные усилия для уточнения назначения конструктов и их формулировок, чтобы данные, собранные в ходе итогового опроса, с высокой долей вероятности были бы достоверными и корректными.<sup>43</sup> Для измерения значений конструктов мы использовали вопросы в стиле шкалы Лайкерта<sup>44</sup>, что позволило провести расширенный анализ.

<sup>43</sup> Мы использовали методологию Черчилля по разработке методов измерения маркетинговых конструктов: Churchill Jr, G. A. "A paradigm for developing better measures of marketing constructs," Journal of Marketing Research 16:1, (1979), 64–73.

<sup>44</sup> McLeod, S. A. (2008). Шкала Лайкерта. Взято из [www.simplypsychology.org/liker-scale.html](http://www.simplypsychology.org/liker-scale.html)

## Методы статистического анализа

- Кластерный анализ. Мы использовали кластерный анализ, чтобы определить профили эффективности поставки программного обеспечения и методики масштабирования, используемые «high performers». Этот подход предполагает, что участники одной группы статистически сходны между собой и отличны от участников других групп по таким параметрам эффективности, как пропускная способность и стабильность: частота развертываний, срок реализации, время восстановления сервиса и доля неудачных изменений. Решение на основе метода Уорда<sup>45</sup> было выбрано по (a) динамики коэффициентов слияния (fusion coefficients), (b) числу участников в каждом кластере (отбрасывались решения с кластерами, содержащими слишком малое число участников) и (c) одномерным F-критериям.<sup>46</sup> Мы использовали метод иерархического кластерного анализа, потому что он имеет высокую объясняющую способность (достаточную для понимания связей между дочерними и родительскими элементами в кластерах), а также потому, что у нас не было отраслевых или теоретических причин иметь заранее определенное количество кластеров. Таким образом, мы хотели, чтобы количество кластеров было определено на основе данных. И наконец, наш набор данных не был слишком большим (метод иерархической кластеризации не подходит для экстремально крупных наборов данных).

45 Ward, J.H. "Hierarchical Grouping to Optimize an Objective Function." *Journal of the American Statistical Association* 58(1963): 236–244.

46 Urich,D., and B. McKelvey. "General Organizational Classification: An Empirical Test Using the United States and Japanese Electronic Industry." *Organization Science* 1, no. 1 (1990): 99–118.

47 Straub, D., Boudreau, M. C., & Gefen, D. (2004). Validation guidelines for IS positivist research. *Communications of the Association for Information systems*, 13(1), 24.

48 <http://www.socialresearchmethods.net/kb/convdisc.htm>

49 <http://www.socialresearchmethods.net/kb/reliable.php>

50 Nunnally, J.C. *Psychometric Theory*. New York: McGraw-Hill, 1978.

51 Chin, W. W. (2010). How to write up and report PLS analyses. In *Handbook of partial least squares* (pp. 655-690). Springer, Berlin, Heidelberg.

52 <http://www.statisticssolutions.com/structural-equation-modeling/>

- Модель измерений. До проведения анализа были определены конструкты с помощью предварительного факторного анализа с использованием ортогонального вращения факторов методом варимакс.<sup>47</sup> Статистические проверки конвергентной и дискриминантной валидности<sup>48</sup> и надежности<sup>49</sup> были подтверждены с помощью средней извлеченной дисперсии (AVE), корреляции, коэффициента надежности Альфа Кронбаха<sup>50</sup> и составной надежности<sup>51</sup>. Конструкты прошли эти тесты, продемонстрировав хорошие психометрические свойства.
- Моделирование структурными уравнениями. Модели структурных уравнений (МСУ)<sup>52</sup> проверялись с помощью метода частичных наименьших квадратов (МЧНК), представляющего собой МСУ на основе корреляции. МЧНК используется в нашем анализе по нескольким причинам. Он не требует предположений о нормальности распределения данных; он хорошо подходит для разведочного и пошагового анализа; он оптимизирован для прогнозирования зависимой переменной (в отличие от проверки соответствия модели данным).<sup>53</sup> Использовался SmartPLS 3.2.8. По отраслям<sup>54</sup> существенное влияние не обнаружено, за исключением розничной торговли ( $p < 0,05$ ), как отмечалось в тексте. По предприятиям (организации с более 5000 сотрудников) обнаружено существенное влияние ( $p < 0,001$ ). По опыту работы существенное влияние не обнаружено. Все пути, представленные на диаграммах МСУ, имеют  $p < 0,001$ , кроме следующих, где  $p < 0,05$ : Непрерывная поставка → Эмоциональное выгорание, Согласование изменений → Эффективность поставки программного обеспечения, Непрерывная интеграция → Непрерывная поставка, Культура → Эффективность поставки программного обеспечения (в главной модели), Внешний поиск → Производительность, Мониторинг → Технический долг, Поддерживаемость кода → Технический долг (в модели производительности).

53 Эти методологические соображения нашли подтверждение в следующих материалах: Chin, W.W. (1998). Issues and opinions on structural equation modeling. *MIS Quarterly*, 22(2), vii-xvi; Gefen, D., Straub, D. W., & Rigdon, E. E. (2011). An update and extension to SEM guidelines for administrative and social science research. *MIS Quarterly*, 35(2), iii-xiv.; and Hulland (1999). Hulland, J. (1999). Use of partial least squares (PLS) in strategic management research: A review of four recent studies. *Strategic Management Journal*, 20(2), 195-204.

54 <http://methods.sagepub.com/reference/encyc-of-research-design/n77.xml>



# БЛАГОДАРНОСТИ

Данное исследование основано на нашей работе и беседах со специалистами по методологиям DevOps, Agile и представителями более широкого технологического сообщества. Выражаем благодарность всем коллегам, которые открыто делились своим опытом и историями, рассказывали об успехах, сложностях и неудачах. На основе этих вводных мы каждый год проводим обзор литературы и составляем вопросы для исследования.

Авторы хотели бы поблагодарить ряд людей за рекомендации и вклад в создание отчета этого года. Все благодарности приведены в алфавитном порядке по типу участия.

Благодарим Адриана Кокрофта (Adrian Cockroft), Сэма Гукенхаймера (Sam Guckenheimer), Джина Кима (Gene Kim) и Келси Хайтауэр (Kelsey Hightower) за общие рекомендации по созданию отчета; как советники вы помогли обозначить ключевые идеи и направления исследования.

За советы по проведению исследования, проверку тем и информацию по измеряемым параметрам нашу благодарность получает команда Engineering Productivity Research компании Google, в частности, Коллин Грин (Collin Green), Сиера Джаспан (Ciera Jaspan), Андреа Найт-Долан (Andrea Knight-Dolan) и Эмерсон Мерфи-Хилл (Emerson Murphy-Hill). Выражаем особую благодарность команде Project Aristotle за ценные аналитические сведения о психологической безопасности и предоставление средств исследования.

Наша работа не увидела бы свет без тщательных проверок профильными экспертами; мы очень благодарны им за время, потраченное на дополнительную вычитку и предоставление информации по измеряемым параметрам.

Нашу благодарность получают Энджи Джонс (Angie Jones), Эшли Макнамара (Ashley McNamara), Бетси Бейер (Betsy Beyer), Бриджит Кромхайт (Brigitte Kromhout), Кортни Кисслер (Courtney Kissler), Дж. Пол Рид (J. Paul Reed), Майк МакГарр (Mike McGarr), Натен Харви (Nathen Harvey), Сет Варго (Seth Vargo) и Ксавье Веласкес (Xavier Velasquez).

Выражаем особую благодарность Дэвиду Ха (David Hu) за поддержку в проведении анализа для исследования и отчета этого года.

Благодарим наших внимательных читателей, которые предоставили нам обратную связь и помогли улучшить отчет: Натен Харви (Nathen Harvey), Том Лимончелли (Tom Limoncelli), Кейти МакКэффри (Caitie McCaffrey), Рейчел Потвин (Rachel Potvin), Кори Куинн (Corey Quinn) и Ксавье Веласкес (Xavier Velasquez).

Авторы хотели бы поблагодарить Шерил Купе (Cheryl Coupé) за внимательность и кропотливое редактирование отчета этого года.

Верстку и дизайн отчета выполнила Шивон Дойл (Siobhán Doyle).



# ОБ АВТОРАХ



Николь Форсгрен (Dr. Nicole Forsgren) возглавляет команду DORA, теперь в рамках Google Cloud. За последние шесть лет она провела самое масштабное на сегодня изучение DevOps в качестве ведущего исследователя отчетов State of DevOps и основного автора книги, удостоенной премии Shingo Publication Award, *Accelerate: The Science of Lean Software and DevOps*. Она работала преподавателем, системным администратором и инженером по эксплуатации. [Работы Николь](#) публиковались в нескольких журналах, рецензируемых экспертами. Николь получила докторскую степень университета Аризоны по управлению информационными системами.

Дастин Смит (Dustin Smith), психолог по человеческому фактору и старший исследователь Google по пользовательским интерфейсам. Он исследует влияние систем и окружающей среды на людей в различных контекстах: разработке программного обеспечения, играх free-to-play, здравоохранении и армии. Его исследования в Google сосредоточены на выявлении аспектов, благодаря которым разработчики программного обеспечения чувствуют себя счастливее и работают более продуктивно. Дастин получил докторскую степень Уичитского государственного университета по психологическим аспектам человеческого фактора.



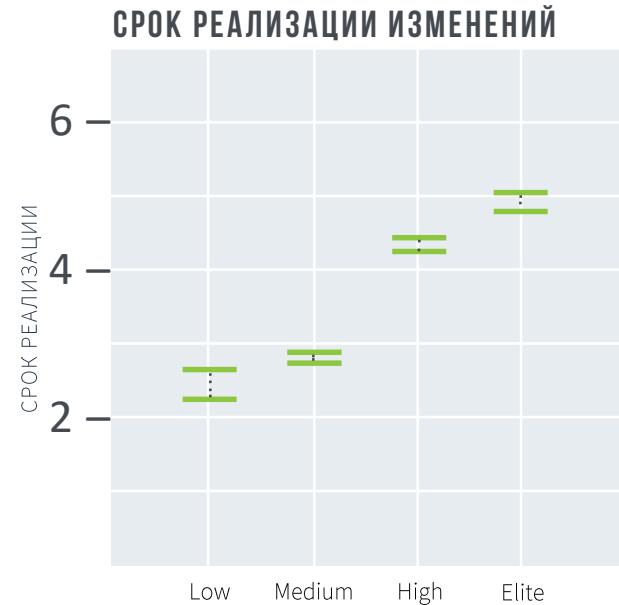
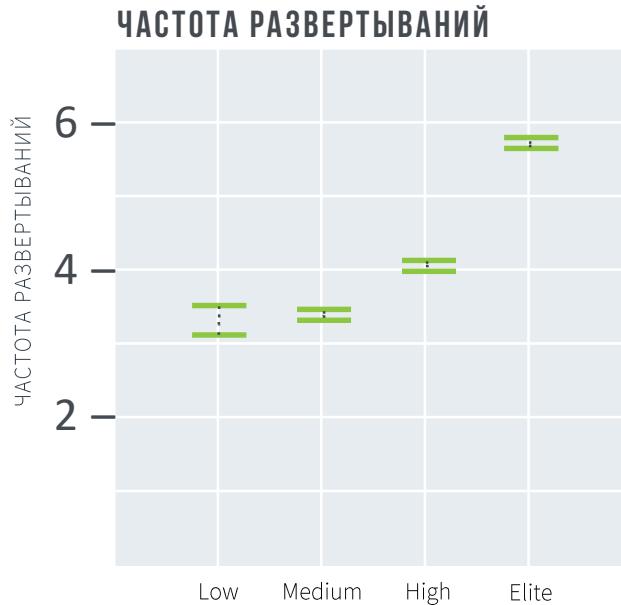
Джез Хамбл (Jez Humble) является соавтором нескольких книг по программному обеспечению, включая удостоенную премии Shingo Publication Award [Accelerate, The DevOps Handbook, Lean Enterprise](#), и удостоенную премии Jolt Award [Continuous Delivery](#). На протяжении своей карьеры он работал с кодом, инфраструктурой и разработкой продуктов в компаниях различных размеров, расположенных на трех континентах. Он трудится в Google Cloud в качестве лидера по продвижению технологий, а также преподает в Калифорнийском университете в [Беркли](#).

Джесси Фразель (Jessie Frazelle), независимый консультант. Работала инженером в нескольких стартапах, а также в Google, Microsoft и GitHub. Джесси знакома с множеством различных практик разработки и организации инфраструктуры благодаря непосредственной работе с соответствующими инструментами (включая Docker и Kubernetes), в том числе в качестве конечного пользователя различных PaaS-решений. Ей нравится смотреть на вещи под новыми углами, переключаться между разработкой инструментов и их применением в производственных средах.



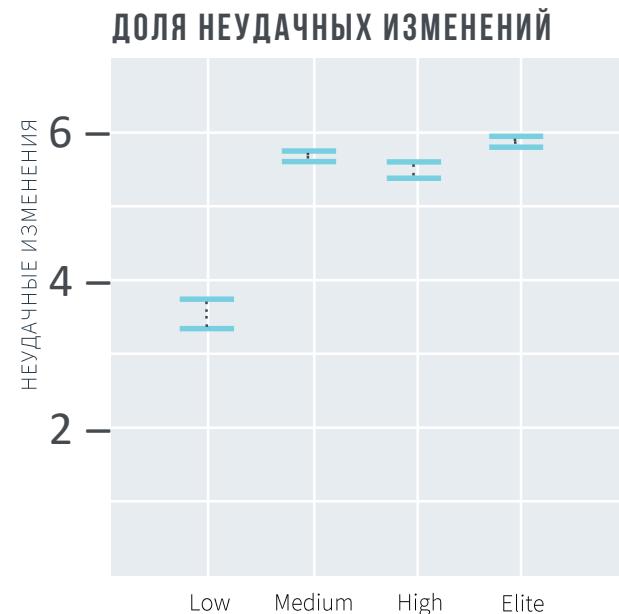
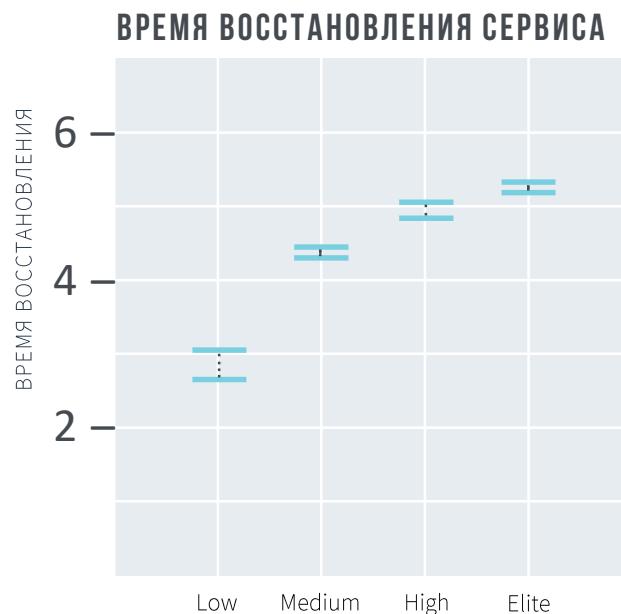


# ПРИЛОЖЕНИЕ А



**ЧАСТОТА РАЗВЕРТЫВАНИЙ**

- 1 = Меньше чем раз в 6 месяцев
- 2 = От одного раза в месяц до одного раза в шесть месяцев
- 3 = От одного раза в неделю до одного раза в месяц
- 4 = От одного раза в день до одного раза в неделю
- 5 = От одного раза в час до одного раза в день
- 6 = По запросу (несколько развертываний в день)



**СРОК РЕАЛИЗАЦИИ ИЗМЕНЕНИЙ**

- 1 = Больше чем 6 месяцев
- 2 = От одного до шести месяцев
- 3 = От одной недели до месяца
- 4 = От одного дня до одной недели
- 5 = Менее чем один день
- 6 = Менее чем один час

**ВРЕМЯ ВОССТАНОВЛЕНИЯ СЕРВИСА**

- 1 = Больше чем 6 месяцев
- 2 = От одного до шести месяцев
- 3 = От одной недели до месяца
- 4 = От одного дня до одной недели
- 5 = Менее чем один день
- 6 = Менее чем один час

**ДОЛЯ НЕУДАЧНЫХ ИЗМЕНЕНИЙ**

- 1 = 76%-100%
- 2 = 61%-75%
- 3 = 46%-60%
- 4 = 31%-45%
- 5 = 16%-30%
- 6 = 0%-15%





# ПРИЛОЖЕНИЕ В

## Стратегии для масштабирования DevOps

- Учебный центр (иногда называется «додзё»). Сотрудники на определенное время отрываются от рабочих обязанностей, чтобы освоить новые инструменты или технологии, практики и даже культурные аспекты, а затем возвращаются в обычную рабочую среду с целью (или надеждой), что освоенный ими способ работы приживется и даже распространится на остальных.
- Центр компетенции. Здесь сосредоточены экспертные знания и навыки и предоставляются консультации другим.
- Проверка концепции, не имеющая продолжения. Проект проверки концепции (РоС), в котором централизованная команда вольна делать все, что она считает лучшим решением, нередко за счет нарушения организационных (и формальных) правил. Однако после этапа РоСэта инициатива сходит на нет.
- Проверка концепции как шаблон. После небольшого проекта проверки концепции (описанной выше) он как шаблон тиражируется в других группах.
- Проверка концепции как отправная точка. Сначала проводится небольшая проверка концепции, после чего полученные в результате РоС знания распространяются на другие группы. Для этого РоС (первая группа РоС или последующие/параллельные группы РоС) разбиваются на части и отправляются другим группам для обмена полученными знаниями и практиками. Этот процесс

можно описать как ротацию, при которой участники РоС интегрируются в другие команды для распространения новых практик и культуры, исполняя роль учителей. Они могут остаться в своих новых группах на неопределенный срок или в течение времени, которое требуется для обеспечения устойчивости новых практик.

- Сообщества по практикам. В организации поощряется формирование групп по интересам: инструменты, языки или методологии. С их помощью происходит обмен знаниями и опытом между участниками, командами и в рамках организации в целом.
- «Большой взрыв». Вся организация переходит на методологии DevOps (в своей трактовке), часто по указанию руководства.
- Подход «снизу вверх», или «Инициатива снизу». Небольшие команды, непосредственно выполняющие работы, объединяют ресурсы для трансформации. В случае успеха они неформально делятся информацией о ней в рамках организации и масштабируют найденные решения без официальной организационной поддержки или ресурсов.
- Смешанный подход. Организация внедряет сразу несколько указанных выше подходов, нередко лишь частично или с нехваткой ресурсов либо не расставив приоритеты для достижения успеха.