УСКОРЯЙСЯ ПО-БЫСТРОМУ

{

Краткое содержание книги
“Ускоряйся” - 1 часть
Николь Форсгрен, Джез Хамбл, Джин Ким

// с комментариями

}

Ускоряйся! Краткое содержание. 1 часть

Доброе) Вот нашел шедевральную книженцию об Айти, ДевОпсе и процессах - “Ускоряйся” (Николь Форсгрен, Джез Хамбл, Джин Ким). Книга о том куда вообще двигаться, чтобы всем стало лучше) Звучит оптимистично и сказочно, но это писали те крутейшие специалисты, что придумали уникальное в своем роде исследование State of DevOps. Они покрутили Айти с разных сторон и рассмотрели такие вещи как: непрерывная доставка ПО, безопасность, каким должен быть лидер, как снизить выгорание и т.д. В общем реально фундаментальный труд. Кому будет полезно: всем, а особенно руководителям и тех.лидам;) Кстати ее посоветовал Барух, когда я искал оригинальные вопросы исследования State of DevOps. «В этой книжке вся механика» - сказал он)

Здесь представлено краткое содержание и основные мысли, слегка разбавленное моими комментариями о суровом энтерпрайзе и оценки всего написанного через призму реальности (они выделены отдельно). Кстати, если какие-то строки вызовут баттхерт - это прямой признак того, что пора что-то менять;)

В введении рассказывается о том, что за исследования проводились. Цель его - покрутить разные аспекты связанные с эффективностью организации (социальные и технические), понять проблематику цифровой трансформации, куда вообще двигаться и как это все померять реальными метриками. Говорится о том, какая классная книга и что она нужна руководству и тех.лидам. Даже Мартин Фаулер (один из отцов Айти ее настоятельно рекомендует, а это прям показатель) Далее книга делится на 3 части: 1 - это что вообще показало это исследование и мысли в какую сторону думать в разрезе своей организации, 2 часть - обоснование исследования (ее можно пока не читать) и 3 - трансформация - я ее еще не прочел, но там немного.

> *Эта книга — своего рода предвидение, в котором так отчаянно нуждаются генеральные директора, финансовые директора и ИТ-директора компаний, если они собираются выжить в этом новом программно ориентированном мире.*
>
> *Любой, кто не читает эту книгу, будет заменен тем, кто ее читал.*
>
> Томас А. Лимончелли, соавтор книги The Practice of Cloud System Administration

> Эта работа — редкий образец глубокой и смелой аналитики, демонстрирующей свежий взгляд на подлинную природу вещей. На основе строгих данных о практике и эффективности DevOps авторы одними из первых объясняют сущность и базовые принципы цифровой трансформации организаций.

Часть 1: Что мы выяснили

Глава 1 - Ускоряйтесь

Цифровая трансформация очень важна. Главное сосредотачиваться на практиках и возможностях (культурных и технологических), а не на оценке зрелости компании.

Чтобы остаться конкурентоспособными и преуспевать на рынке, организации должны ускорить:

- доставку товаров и услуг, чтобы порадовать своих клиентов
- взаимодействие с рынком для выявления и понимания потребительского спроса
- предвидение измерений в области регулирования и соблюдение требований, которые влияют на их системы
- реакцию на потенциальные риски, такие как угрозы безопасности или изменения в экономике

Подчеркивается ценность внедрения DevOps.

Подводя итог, в 2017 году мы обнаружили, что по сравнению с низкоэффективными компаниями лидеры отрасли показывают следующие результаты:

- в 46 раз более частое развертывание кода;
- в 440 раз меньшее время выполнения от коммита до развертывания;
- в 170 раз меньшее среднее время восстановления после простоя;
- в 5 раз более низкий показатель отказов изменений (1/5, скорее для того, чтобы изменение не состоялось).

Глава 2 - Измерение эффективности

Измерение эффективности в области разработки - довольно нетривиальная задача и все метрики, что были ранее (количество строк кода, стори-поинты в Agile, нагрузка на разработчиков) не очень для этого подходят.

Успешное измерение эффективности должно:

- Сосредоточено на глобальном результате, чтобы команды не сталкивались друг с другом. (не должно быть так, чтобы разрабов награждали за скорость, а эксплуатацию - за стабильность)
- Сфокусировано на конечном результате, а не на промежуточных. Оценка не должна вознаграждать за работу, которая не приносит ценности и не помогает достичь организационных целей

Авторы выдвинули 4 метрики эффективности доставки ПО:

- **Время выполнения изменений (Lead Time for Change)**
  Это время внедрения фичи на прод. Т. к само время разработки весьма вариативно, то тут имеется в виду время внедрения фичи: от коммита в dev и до появления фичи на проде (после тестирования, согласования и т.д)
- **Частота развертывания (Deployment Frequency)**
  Т.к размер пакета изменений в коде тоже довольно относительный и его сложно померять, то предлагается измерять насколько часто изменения накатываются в прод.
- **Среднее время восстановления (Mean Time to Restore - MTTR).**
  Сколько времени нужно, чтобы починить упавший сервис.
- **Процент отказов при внесении изменений (Change Fail Percentage)**
  Процент брака при внесении изменений.

Используя кластерный анализ, авторы разделили организации на 3 категории: лидеры, средние и отстающие.

Я: Кстати, кому интересно - это отчет называется State of DevOps, довольно интересный.

Эффективность доставки ПО в 2016 году

2016	Лидеры	Средние	Отстающие
Частота развертывания	По запросу (множественные развертывания в течение дня)	От одного раза в неделю до одного раза в месяц	От одного раза в неделю до одного раза в полгода*
Время внесения изменений	Менее одного часа	От одной недели до одного месяца	От одного месяца до полугода
Среднее время восстановления	Менее одного часа	Менее одного дня	Менее одного дня*
Процент отказов при изменениях	0–15%	31–45%	16–30%

Таблица 2.3

Эффективность доставки ПО в 2017 году

2017	Лидеры	Средние	Отстающие
Частота развертывания	По запросу (множественные развертывания в течение дня)	От одного раза в неделю до одного раза в месяц	От одного раза в неделю до одного раза в полгода*
Время внесения изменений	Менее одного часа	От одной недели до одного месяца	От одного месяца до полугода
Среднее время восстановления	Менее одного часа	Менее одного дня	От одного дня до одной недели
Процент отказов при изменениях	0–15%	0–15%	31–45%

* Компании с низкой эффективностью имели худшие результаты (на статистически значимом уровне), но их средние показатели были близки к показателям участников со средними результатами.

Далее приводятся тренды и организации с высокими показателями их как правило еще улучшают, а с низкими - периодически деградируют. Был замечен интересный эффект, что у средних групп порой возрастал процент отказов. Это видимо связано с перестройкой архитектуры и внедрением DevOps-практик, т, е просто временный переходный период и процесс адаптации.

Я: про это есть в книжке:
https://sre.google/resources/practices-and-processes/enterprise-roadmap-to-sre/

Далее в книге "Ускоряйся" везде приводятся диаграммы на что влияет каждая характеристика. Серым - обозначены конструкции (конструкции - это выверенные статистикой характеристики), а стрелочками показано позитивное влияние.

Рис. 2.4. **Влияние эффективности доставки ПО**

Еще важные мысли в этой главе:

- Определение того, какое ПО является стратегическим, а какое нет, и управление ими соответствующим образом имеют огромное значение.
  Нестратегическое ПО: большинство программ, которые используются для выполнения работы (офисные программы, программы начисления зп и т.д)
  Нестратегическое ПО должно приобретаться (в основном)
  Я: Например сервисы, которые не приносят прямой бизнес-ценности: сервисы по обмену с ФНС, ФССП, ПФР и т.д - нестратегические
  А сервисы, связанные с ДБО ФЛ, ДБО ЮЛ - стратегические, хотя и присутствует вендорное ядро от BSS. Многие доработки делаются своими силами.
- Важно осмотрительно использовать приведенные выше характеристики для оценки эффективности. В организациях с культурой обучения они невероятно сильны. Но в патологических и бюрократических организациях оценка используется как форма контроля и люди скрывают информацию, которая бросает вызов существующим правилам, стратегиям и структурам власти.

Глава 3 - Измерение и изменение культуры

Это практически прописная истина в кругах DevOps: культура имеет огромное значение.

Я: Часто говорят, что DevOps - это лишь на 30% про технологии и 70% - про людей и процессы. Не раз сталкивался с людьми, которые отвергают как культуру, так и технологии. Что прискорбно, т.к Айти - как раз та область, где очень важно и то и другое, как впрочем и в любой современной организации. Важно развитие, важна командная и продуктивная работа, совместная работа отделов и людей. Т.к чтобы сделать полноценный продукт важны роли всех: аналитиков, разработчиков, эксплуатации, безопасности и т.д.

Организационная культура может существовать на 3-х уровнях:

- **Основные положения**
  Это то, что формируется с течением времени, это сложно описать, это как общепринятые вещи.
- **Ценности**
  Ценности, которые являются более видимыми для членов группы, могут обсуждаться, оспариваться. Они влияют на групповые взаимодействия, устанавливают социальные нормы, правила, которые формируют действия людей. Именно этот уровень обычно ассоциируется с понятием "культура"
- **Артефакты**
  Это письменные заявления или убеждения, связанные с миссией компании, формальные процедуры.

Основываясь на дискуссиях в кругах DevOps и важности организационной культуры на втором уровне, авторы решили выбрать для классификации организаций в разрезе культуры модель социолога Рона Веструма.

Классификация организационных культур по Веструму:

- **Патологические (ориентированные на власть)**
  Организации характеризуются атмосферой страха и угрозы. Люди часто копят или скрывают информацию по политическим причинам или искажают ее, чтобы выставить себя в лучшем свете.
- **Бюрократические (ориентированные на правила)**
  Организации защищают отделы. Те, кто в департаменте, хотят сохранить свою "территорию", настаивают на собственных правилах и обычно делают все по правилам - своим правилам.
- **Производительные (ориентированные на результат)**

Следующая идея Веструма состояла в том, что организационная культура влияет на то, как информация распространяется внутри организации.

Веструм приводит три характеристики хорошей информации:

- Она дает ответы на вопросы, которые востребованы тем, кто их задает.
- Она своевременна
- Она представлена таким образом, что может быть эффективно использована тем, кто ее получает.

Я: читая книгу или это краткое содержание - постарайтесь соотнести ее на свои кейсы и ответить на вопросы.

Дополнительное открытие Веструма состояло в. том, что определение типа организационной культуры предсказывает результаты работы.

Типология организационной культуры по Веструму

Патологические (ориентированные на власть)	Бюрократические (ориентированные на правила)	Производительные (ориентированные на результат)
Низкий уровень сотрудничества	Средний уровень сотрудничества	Высокий уровень сотрудничества
«Гонцов» с плохими новостями «пристреливают»	«Гонцов» игнорируют	«Гонцов» обучают
Уклонение от ответственности	Узкая область ответственности	Разделение рисков
Горизонтальные связи не допускаются	Горизонтальные связи допускаются	Горизонтальные связи поощряются
Ошибки ведут к поиску козла отпущения	Ошибки регулируются правилами	Ошибки ведут к исследованиям
Инновации подавляются	Инновации приводят к проблемам	Инновации внедряются

Я: под гонцами тут понимается восприятие плохих новостей: в производительных организациях, например, в случае аварий составляют постмортемы (это детальное описание аварий и как исправили ситуацию) и инструкции по устранению, а затем делают доработки и чинят.
В общем, учатся на своих ошибках.

Важные мысли в этой главе:

- В организациях с производительной культурой люди сотрудничают более эффективно, и в них присутствует более высокий уровень доверия по всей организационной иерархии.
- Производительная культура подчеркивает миссию - акцент, который позволяет вовлеченным людям оставить в стороне свои личные проблемы, а также отраслевые проблемы, которые так очевидны в бюрократических организациях. Миссия первична.
- Созидательность поощряет равные условия, в которых иерархия играет не такую важную роль.

- Бюрократия - это не обязательно плохо. Цель бюрократии - обеспечить справедливость путем применения правил к управленческому поведению. Правила позволяют исключить дискриминацию, устраняя произвол, а также правила могут агрегировать опыт управления.
- Теория Веструма утверждает, что организации с лучшим информационным потоком функционируют более эффективно.
- Хорошая культура требует доверия и сотрудничества между людьми по всей организации.
- Более высокая организационная культура может свидетельствовать о более качественном принятии решений. В команде с таким типом культуры не только больше информации доступно для принятия решений, но и эти решения легче отменить, если они окажутся неверными, потому что команда, скорее всего, будет открытой и прозрачной, а не закрытой и иерархической. Команды с такими культурными нормами, скорее всего, будут обладать лучшей атмосферой для людей, поскольку проблемы будут быстрее обнаруживаться и решаться.
- Для современных организаций, которые надеются преуспеть перед лицом все более быстрых технологических изменений, важны как устойчивость, так и способность к инновациям через реагирования на эти изменения.
- Изменить культуру - это не сначала изменить то, как люди думают, а вместо этого начать с изменения того, как люди себя ведут и что они делают.

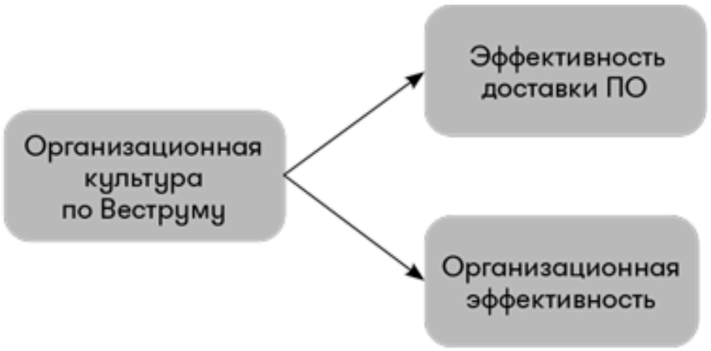


Рис. 3.2. **Результаты внедрения организационной культуры по Веструму**

Рис. 3.3. **Движущие факторы организационной культуры по Веструму**

Глава 4 - Технические практики

Технические практики играют очень важную роль в достижении результатов, особенно практика непрерывной доставки (Continuous Delivery) Далее в главе обсуждаются исследования, чтобы измерить НД как возможность и оценить ее влияние на эффективность доставки ПО, организационную культуру и другие показатели результатов, такие как выгорание и боль развертывания.

Непрерывная доставка - это набор возможностей, которые позволяют нам безопасно, быстро и устойчиво внедрять изменения всех видов: функции, изменения конфигурации, исправления ошибок, эксперименты, - в производство или непосредственно пользователям.

В основе НД лежат пять ключевых принципов:

- **Качество сборки**
  Очень важно обнаружить и устранить любые проблемы, как можно дешевле и быстрее.
- **Работа небольшими партиями**
  Позволяет избегать работы с нулевой или отрицательной ценностью для организаций.
- **Компьютеры выполняют повторяющиеся задачи; люди решают проблемы**
  Одна из важных стратегий снижения затрат на выпуск изменений заключается в инвестировании в упрощение и автоматизацию повторяющейся работы, которая занимает много времени. Таким образом мы освобождаем людей для более ценной работы по решению таких задач, как улучшение структуры наших систем и процессов в ответ на обратную связь.
  Я: К тому же, человек плох в автоматических рутинных действиях, во всяком случае гораздо хуже компьютера и более склонен к ошибкам и усталости.
- **Неустанно проводить постоянные улучшения**
  Наиболее важной характеристикой высокоэффективных команд является то, что они никогда не бывают удовлетворены: они всегда стремятся стать лучше. В компаниях-лидерах улучшение является частью ежедневной работы каждого сотрудника.
- **Каждый несет ответственность**
  В бюрократических организациях команды, как правило, сосредоточены на целях своего подразделения, а не на организационных целях. Однако важно тесное сотрудничество между всеми участниками процесса доставки ПО.

Чтобы осуществлять непрерывную доставку ПО, необходимо создать следующие основы:

- **Комплексное управление конфигурацией**
  Должна быть возможность обеспечить среду разработки и создавать, тестировать, развертывать наше ПО полностью автоматизированным способом исключительно из информации, хранящейся в системе контроля версий. Любые изменения в среде или ПО, которое на ней работает, должны применяться с использованием автоматизированного процесса из системы управления версиями. Это все еще оставляет место для ручных утверждений концепций, но как только они утверждены - они должны применяться автоматически.
- **Непрерывная интеграция (Continuous Integration)**
  Высокоэффективные команды сливают код в короткоживущие ветки (менее одного дня) и часто сливают их в магистральную ветку (master, main). Каждое изменение запускает процесс сборки и тестирования качества продукта. Автоматические модульные и приемочные тесты должны применяться при каждой фиксации кода в системе контроля версий, чтобы дать разработчикам быструю обратную связь. Никто не должен утверждать, что работа сделана, пока необходимые тесты не будут пройдены.

Реализация непрерывной доставки означает создание многочисленных циклов обратной связи для обеспечения того, чтобы высококачественное ПО доставлялось пользователям чаще и надежнее.

Исследования, проводимые авторами показали, что команды, хорошо справляющиеся с непрерывной доставкой, сильнее идентифицировали себя организацией в которой они работали - а это ключевой показатель организационной эффективности.

Я: если коротко - это о том, насколько вы гордитесь работать в компании, далее эта тема будет подробнее раскрыта.

Предоставляя разработчикам инструменты для обнаружения возникающих проблем, время и ресурсы для инвестирования в их развитие, а также полномочия для немедленного устранения проблем, мы создаем среду, в которой разработчики принимают ответственность за глобальные результаты, такие как качество и стабильность. Это оказывает положительное влияние на групповые взаимодействия и деятельность организационной среды и культуры членов команды.

С помощью непрерывной доставки команды смогут:

- Выполнять развертывания по требованию.
- Получать быструю обратную связь и корректировать свои действия в соответствии с ней.

На непрерывную доставку позитивно влияют две вещи:

- слабосвязанная, хорошо инкапсулированная архитектура
- возможность команд выбирать подходящие инструменты

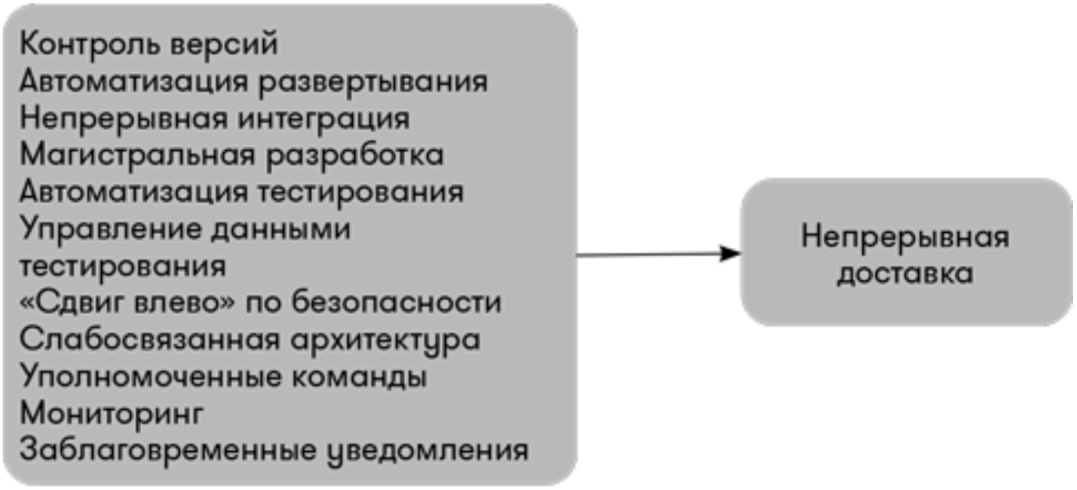


Рис. 4.1. **Движущие факторы непрерывной доставки**

Команды, которые хорошо справляются с непрерывной доставкой, достигали следующих результатов:

- сильная идентификация с организацией, в которой они работают
- более высокие уровни эффективности доставки ПО (время выполнения, частота развертывания, время восстановления)
- более низкий процент отказов при изменениях
- производительная культура, ориентированная на результат
- меньше боли развертывания
- снижение командного выгорания
- выше качество и производительность приложений
- меньше процент времени, затраченный на доработку или незапланированную работу
- меньше процент времени, затраченного на работу с ошибками, выявленными конечными пользователями

Рис. 4.2. **Влияние непрерывной доставки**

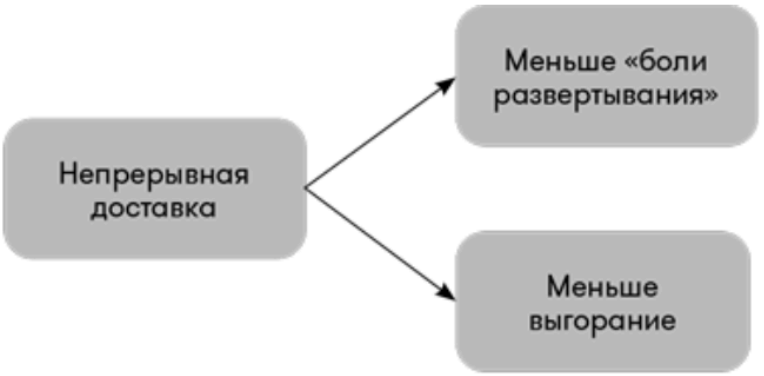


Рис. 4.3. **Непрерывная доставка делает работу более устойчивой**

Незапланированная работа и доработка являются полезными индикаторами качества, потому что они представляют собой невозможность встроить качество в наши продукты.

В книге The Visible Ops Handbook незапланированная работа описана как разница между тем, как если “обратить внимание на сигнал низкого уровня топлива в автомобиле или остаться без топлива на шоссе”. В первом случае организация может устранить проблему плановым образом, без особой срочности или нарушения других запланированных работ. Во втором случае они вынуждены решать проблему в очень срочном порядке, часто задействуя все ресурсы, - представьте, что шесть инженеров должны бросить все и бежать по шоссе с полными газовыми баллонами, чтобы заправить застрявший грузовик.

Еще важные и интересные мысли из этой главы:

- Сохранение конфигурации системы и приложения в системе контроля версий более тесно связано с эффективностью доставки ПО, чем сохранения кода приложения в системе контроля версий. Конфигурация обычно считается второстепенной проблемой по сравнению с кодом приложения в управлении конфигурацией, но исследование показало, что это ошибочное представление.
- Разработчики в основном создают и поддерживают приемочные тесты.
- Наличие автотестов, созданных и поддерживаемых QA-командой или аутсорсинговой стороной не коррелирует с ИТ-эффективностью.
  Код проще тестировать, когда разработчики пишут тесты и они вкладывают больше сил в их исправление и поддержание. Но это не значит, что мы должны избавиться от тестировщиков. Тестировщики играют важную роль в жизненном цикле доставки ПО, выполняя ручное тестирование: исследовательское, приемочное и тестирование юзабилити - а также помогая разработчикам создавать и развивать наборы автотестов.
- Магистральная разработка (trunk development), в отличие от разработки на долгоживущих ветках, коррелировала с более высокой эффективностью доставки. У успешных команд было: менее 3-х ветвей, живущих менее 1 дня без замораживания веток. Однако GitHub Flow подходит для опенсорсных проектов, где участники работают с кодом, когда есть время.
- Реализация практик непрерывной доставки требует переосмысления всего - от того, как команды работают, до того как они взаимодействуют друг с другом, какие инструменты и процессы они используют.

Глава 5 - Архитектура

Исследования показали, что высокая эффективность возможна со всеми видами систем при условии, что системы и команды, которые их строят и поддерживают, слабо связаны. Это ключевое архитектурное свойство позволяет командам легко тестировать и развертывать отдельные компоненты или службы, даже когда организация и количество систем, которыми она управляет, растут, т.е это позволяет организациям увеличивать свою производительность по мере масштабирования.

Авторы исследовали различные виды систем от стартапов до по для мейнфреймов. В целом исследование показало, что нет значимой корреляции между типом системы и эффективностью доставки ПО, но самые низкие показатели были при разработке ПО аутсорсом.

Важно инвестировать в собственные возможности для создания и развития основных, стратегических программных продуктов и услуг, которые обеспечивают ключевое отличие вашего бизнеса.

Такие характеристики архитектуры как тестируемость и развертываемость важны для создания высокой эффективности. Для достижения этих характеристик системы разработки (сервисы) слабо связаны, т.е могут быть изменены и проверены независимо друг от друга.

Самый большой вклад в непрерывную доставку состоит в том, могут ли команды:

- вносить масштабные изменения в разработку своей системы без кого-либо вне команды
- вносить масштабные изменения в разработку своей системы, независимо от того, внесли ли изменения в собственные системы другие команды, и не создавая при этом значительный объем работ для других команд.
- завершить свою работу без коммуникации и координации с людьми вне своей команды
- осуществить развертывание и выпуск продукта или услуги по требованию, независимо от других сервисов, от которых они зависят
- провести большую часть тестирования по требованию, не требуя интегрированной среды
- выполнить развертывание в обычное рабочее время с незначительным временем простоя

В общем архитектура и команды должны быть слабо связанными. Для этого команды должны быть кроссфункциональными (со всеми необходимыми навыками)

То, что архитектура и команды должны быть слабосвязанными исходит и из закона Конвея (и обратного закона Конвея), который говорит о том, что организации проектируют системы, которые копируют структуру коммуникаций в этой организации и наоборот.

Архитектурные подходы, позволяющие достигать этого включают использование API-интерфейсов, ограниченных контекстов, декомпозицию больших областей на более мелкие и т.д.

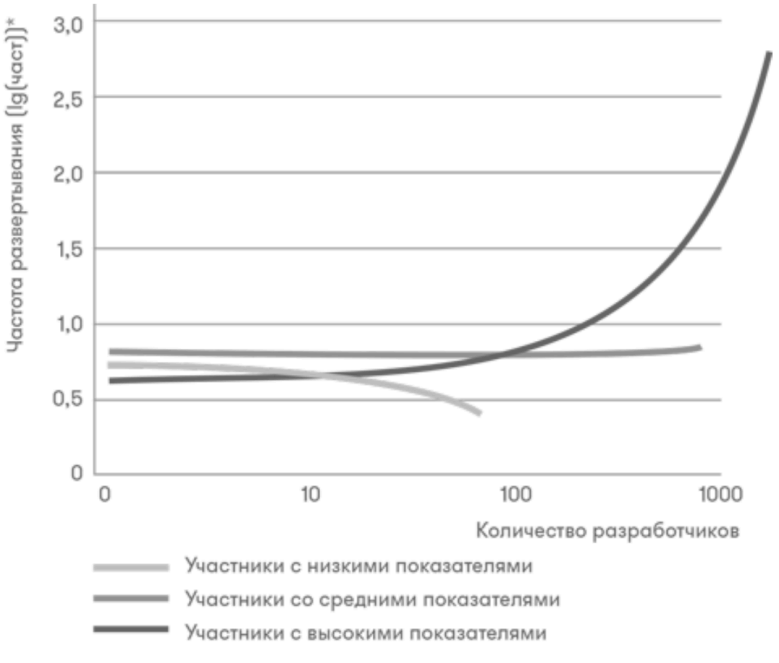
Конечно, DevOps - это прежде всего лучшее взаимодействие между командами, и мы не хотим сказать, что команды не должны работать вместе. Цель слабосвязанной архитектуры состоит в том, чтобы гарантировать, что каналы коммуникации не перегружены принятием мелких решений и мы можем вместо этого использовать эти каналы для обсуждения общих целей более высокого уровня и способов их достижения.

Слабосвязанная, хорошо инкапсулированная архитектура дает:

- лучшую эффективность доставки (скорость, стабильность, меньшее выгорание)
- возможность существенно увеличить размер инженерной организации и повысить производительность линейно или даже больше

По мере увеличения числа разработчиков исследования показали следующее:

- участники из группы с низкими показателями эффективности сокращают частоту развертывания кода (из за возросшего числа коммуникаций и накладных расходов на интеграцию)
- участники из группы со средними показателями развертывают код с постоянной частотой
- участники из групп с высокими показателями значительно увеличивают частоту развертывания кода



* Чем выше, тем чаще: 1 = ежедневно.

Еще важные мысли из этой главы:

- Во многих организациях существует стандартный набор инструментов, которыми могут пользоваться инженеры. С одной стороны стандарты - это хорошо, т.к это уменьшает сложность среды, упрощает использование навыков к заданным технологиям, упрощает лицензирование и т.д.
- С другой стороны, важно давать инженерам возможность использовать подходящие и удобные инструменты. Исследование показало, что выбор подходящих инструментов увеличивает эффективность доставки ПО.
  Я: К тому же выбранные инструменты, потом могут войти в стандарт.
- Когда предоставленные инструменты действительно будут облегчать жизнь инженерам, которые их используют, они примут их добровольно. Это гораздо лучше, чем заставлять их использовать инструменты, которые выбраны для удобства других заинтересованных сторон. Фокус на удобстве использования и удовлетворенности клиентов также важен при выборе или создании инструментов для внутренних клиентов.
- Архитекторы должны фокусироваться на инженерах и результатах, а не на инструментах или технологиях.
- Какие инструменты или технологии вы используете, не имеет значения, если люди, которые должны их использовать, ненавидят работать с ними или если они не достигают конечных результатов и не дают нужных нам моделей поведения.
- Команды, которые встраивают системы безопасности в свою работу, также лучше справляются с непрерывной доставкой.

Глава 6 - Интеграция информационной безопасности в жизненный цикл доставки

Инфобез также важная часть DevOps. Зачастую команды безопасности плохо укомплектованы. Джеймс Уиккет, руководитель исследований в Signal Sciences, приводит соотношение: 1 безопасник на 10 админов и 100 разработчиков в крупных компаниях. При этом они обычно участвуют только в конце жизненного цикла доставки ПО, когда уже часто мучительно больно и дорого вносить изменения, необходимые для повышения безопасности. Кроме того, многие разработчики не знают об общих рисках безопасности, таких как OWASP, и как их предотвратить. Когда команды "сдвигаются влево" по информационной безопасности, т.е когда они встраивают ее в процесс доставки ПО, вместо того, чтобы делать отдельной фазой, которая происходит после процесса разработки, это положительно влияет на их способность осуществлять непрерывную доставку.

Я: Например, если контейнеризовать приложение в docker-образ, упаковав его в immutable-контейнер, то все проверки можно сделать до деплоя в прод. Или, к примеру, если обнаружили уязвимость в Линуксе, то исправить ее на проде, на котором развернуто множество сервисов - крайне сложно, в то время как в докер-образе достаточно лишь изменить одну строчку с базовым образом.

Информационная безопасность должна быть неотъемлемой частью всего жизненного цикла доставки ПО и должна предоставлять удобные инструменты для проверки безопасности и предоставлять тестирование безопасности в рамках автотестов, не замедляя процесс разработки.

То, что мы видим здесь, - это переход от ситуации, когда команды безопасников сами проводят проверку безопасности, к предоставлению разработчикам средств для создания встроенной безопасности. Это отражает две реальности:

- Гораздо легче убедиться, что люди, создающие ПО, делают все правильно, чем проверять почти заверщенные системы и функции, чтобы найти значительные архитектурные проблемы и ошибки, которые требуют существенной переработки.
- Команды инфобеза просто не имеют возможности проводить проверки безопасности при частых развертываниях. Привлечение безопасников на протяжении всего процесса разработки также положительно влияет на коммуникационные и информационные потоки, а это и есть взаимовыгодная и основная цели DevOps.

В целях сокращения времени и затрат на обеспечение безопасности - удобно встраивать ее в Платформу (например, внутреннее облако)

Глава 7 - Управленческие практики при работе с ПО

Одна из практик управления поставкой ПО - бережливое производство (Lean Development). Изначально придумана на заводе Toyota. Авторы рассматривают бережливое производство с помощью 3-х компонентов:

- Ограничение незавершенного производства (НЗП) и использование этих ограничений для улучшения процессов и увеличения производительности
- Создание и поддержка визуальных дисплеев, показывающих ключевые показатели качества и производительности, а также текущий статус работы (включая ошибки), доступ к этим дисплеям для инженеров и руководителей и согласование данных показателей с операционными целями.
- Использование на ежедневной основе данных из инструментов мониторинга инфраструктуры и эффективности приложений для принятия бизнес-решений.

Ограничения НЗП сами по себе не сильно влияют на эффективность доставки. Однако наблюдается сильный эффект, когда они объединены с средствами визуализации и имеют обратный цикл связи от систем мониторинга производства до команд доставки или бизнеса. Когда вся эта информация на виду, включая ошибки - команды стремятся улучшить свои результаты.

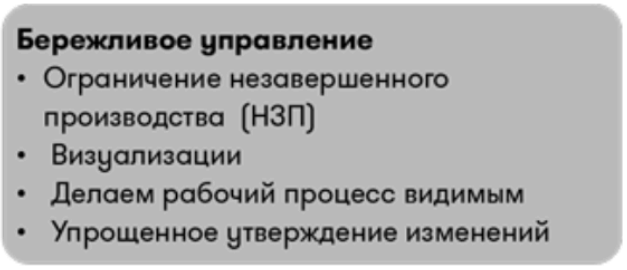


Рис. 7.1. **Компоненты бережливого управления**

Рис. 7.2. **Влияние практик бережливого менеджмента**

Бережливый менеджмент повышает эффективность доставки, оказывает положительное влияние на культуру и снижает выгорание.

В крупных организациях мы часто видим процессы управления изменениями, которые занимают дни или недели, требуя, чтобы каждое изменение было рассмотрено консультативным советом по изменениям (CAB - change advisory board), внешним по отношению к команде, в дополнение к экспертизам на уровне команды, таким как формальный процесс проверки кода.

Я: У нас обычно это называется, если говорить простым языком - согласования-пересогласования, Заявко-Опс или футбол служебками. Да есть ситуации, несомненно требующие согласования, но их значительно меньше, к тому же согласования и схемы можно масштабировать и упрощать. Например, если деплой идет в кластер, то, не имеет смысла каждый раз согласовывать все хосты единично, достаточно согласовать деплой в кластер. Или, к примеру, излишние согласования для разработчиков в DEV-среде могут быть сведены к минимуму.

Проведенные исследования показали, что утверждение только изменений с высоким риском не коррелирует с эффективностью доставки ПО. Команды, которые сообщили об отсутствии процесса утверждения или использовали внутреннюю экспертизу, достигали более высокой эффективности доставки ПО. Наконец, команды, которым требовалось утверждение со стороны внешнего органа, показали более низкую эффективность.

Совсем без утверждений - плохо, но процесс утверждений должен быть максимально простым и прозрачным. Авторы советуют использование упрощенного процесса утверждения изменений, основанного на экспертной оценке, такой как парное программирование или проверка кода внутри команды, в сочетании с конвейером развертывания для обнаружения и отклонения плохих изменений. Этот процесс можно использовать для всех видов изменений, включая изменения кода, инфраструктуры и баз данных.

Я: Понятно, что в энтерпрайзе обязан быть определенный процесс утверждений в силу требований регуляторов, например.

А как насчет разделения ответственности?

В регулируемых отраслях разделение ответственности часто требуется либо буквально в формулировках регламента (например, в случае PCI DSS), либо аудиторами. Однако реализация этого контроля не требует CAB или отдельной рабочей группы. Существуют два механизма, которые могут быть эффективно использованы для того, чтобы соответствовать букве и духу этого контроля.

Во-первых, когда какое-либо изменение зафиксировано, кто-то, кто не участвовал в создании изменения, должен просмотреть его либо до, либо сразу после фиксации в системе контроля версий. Это может быть кто-то из той же команды. Этот человек должен утвердить изменение, записав свое одобрение в систему контроля версий, путем принятия мердж-реквеста или через инструмент конвейера развертывания, утвердив ручной этап, сразу после фиксации изменения.

Во-вторых, изменения должны применяться только к производству с использованием полностью автоматизированного процесса, который является частью конвейера развертывания. Т.е никакие изменения не могут быть внесены в производство, если они не зафиксированы в системе управления версиями, проверены стандартным способом сборки и тестирования, а затем развернуты с помощью автоматизированного процесса, запускаемого конвейером развертывания (пайплайна). В результате применения пайплайна аудиторы будут иметь полную запись о том, какие изменения были применены к каким средам, откуда они берутся в системе контроля версий, какие тесты и проверки были выполнены и кто и когда их одобрил. Таким образом конвейер развертывания особенно ценен в контексте строго регулируемых отраслей или отраслей, для которых безопасность критически важна.

Еще важные мысли:

- Какова вероятность, что внешний орган просмотрит все изменения, сделанные в системе разработчиками?
- Есть ситуации, когда люди вне команды могут осуществлять эффективное управление рисками изменений. Однако это скорее роль руководства, чем собственно проверка изменений. Такие команды должны контролировать эффективность доставки и помогать командам улучшать ее, внедряя практики, которые повышают стабильность, качество и скорость, такие как непрерывная доставка и методы бережливого управления.

Глава 8 - Разработка продукта

Не все используют Agile как надо, к тому же зачастую обработка обратной связи от клиентов рассматривается в десятую очередь. Напротив, и концепция бережливой разработки продукта, и движение бережливого запуска делают акцент на тестировании дизайна и бизнес-модели вашего продукта, собирая частую обратную связь.
Бережливая разработка продукта:

- **Работа небольшими партиями**
  Команды разделяют продукты и функции на небольшие партии, которые могут быть завершены менее чем за неделю и выпущены частыми релизами, включая использование минимально жизнеспособных продуктов (MVP - minimum viable product) Работа небольшими партиями также позволяет делать A/B-тестирование.
- **Визуальный менеджмент**
  Есть ли у команд хорошее понимание всего потока работы от бизнеса до клиентов и есть ли у них прозрачность в этом потоке, включая статус продуктов и функций.
- **Сбор и внедрение обратной связи от клиентов**
  Собирается ли регулярно обратная связь от клиентов и включается ли сбор обратной связи в разработку самого продукта (в пайплайны)
- **Командные эксперименты**
  Есть ли у группы разработчиков полномочия создавать и изменять спецификации в рамках процесса разработки без необходимости утверждения.

Обратная связь очень важна на протяжении всего процесса разработки. Это позволяет команде разработчиков собирать важную информацию, которая помогает скорректировать направление разработки. Таким образом обратная связь позволяет командам интегрировать исследования опыта пользователей в разработку и доставку продукта. Но, если команде разработчиков не разрешается без какого-то официального одобрения какого-либо внешнего органа изменять требования или спецификации в ответ на то, что они обнаружили, их способность к инновациям резко тормозится.

Получение обратной связи от клиентов включает в себя несколько методов: регулярный сбор показателей удовлетворенности клиентов, активный поиск информации о мнении клиентов относительно качества продуктов и функций и использование этой обратной связи для информирования о дизайне продуктов и функций. Важно также и то, в какой степени команды действительно имеют полномочия реагировать на эту обратную связь.

Я: Например, представьте мобильное приложение банка. Можно собирать обратную связь с отзывов о приложении в маркетплейсе, соц.сетях, на основе бизнес-метрик или используя RUM (real user monitoring), когда на мобильное приложение вешается агент, собирающий пользовательские метрики: кто какими режимами пользуется, какие функции крайне популярны, а какие практически не используются, сколько времени происходит загрузка и выполнение функций и т.д.

Авторы не предлагают вам освободить своих разработчиков и позволить им работать над любыми идеями, которые им нравятся. Чтобы быть эффективным, экспериментирование должно сочетаться с другими возможностями, которые мы здесь измеряем: работа небольшими партиями, создание видимого для всех потока работы над всем процессом доставки и включение обратной связи в разработку продуктов.

Бережливое управление продуктом ведет к более эффективной доставки ПО и наоборот: эффективная доставка ПО предсказывает методы бережливого управления.

Рис. 8.2. **Влияние бережливого продукт-менеджмента**

Глава 9 - Как обеспечить устойчивость работы

Страх и тревога, которые испытывают инженеры и технический персонал, когда они запускают код в прод, могут многое рассказать нам об эффективности доставки ПО. Мы называем это “болью развертывания”, и ее важно измерить, т.к она подчеркивает трения и разногласия, которые существуют между разработкой, эксплуатацией, тестированием ПО и работой для поддержания его работоспособности. Именно здесь разработка встречается с эксплуатацией, и именно здесь существует наибольший потенциал для различий: в среде, в процессе и методологии, в мышлении и даже в словах, которые команды используют для описания своей работы.

Исследования показали, что там, где развертывание кода наиболее болезненно, вы найдете самую низкую эффективность доставки ПО, эффективность организации и культуру.

В Микрософт до внедрения практик и дисциплины непрерывной доставки уровень удовлетворенности команды Bing составлял 38%, после внедрения - 75%

Плохо, когда эксплуатация не имеет представления, о развертываемом продукте, также как и когда группы разработчиков и тестирования не имеют представления о том, что такое развертывание, когда у них нет представления о инфраструктуре. Барьеры, которые скрывают работу по развертыванию от разработчиков, редко бывают полезными, потому что они изолируют разработчиков от последствий из работы.

Я: не раз слышал от эксплуатации: давайте вы нам киньте, что разворачивать и инструкцию, а мы развернем. Обычно это приводило к проблемам в развертывании, где-то что-то забыли, что-то развернули не так, чего-то не хватило в инструкции или вообще развернули неактуальную версию. В результате появлялись ошибки и лишнее затраченное время нескольких специалистов. Плюс разработчику важно понимать, что сервис, разработанный им ведет себя корректно, это непосредственно влияет на удовлетворенность от выполненной работы.

Что снижает боль развертывания:

- автоматизация развертывания
- автотесты
- магистральная разработка
- сдвиг влево по безопасности
- слабосвязанные архитектуры
- использование системы контроля версий для кода, конфигурации и инфраструктуры

Если развертывание должно выполняться в нерабочее время - это признак архитектурных проблем, которые должны быть решены. Вполне возможно - при

наличии достаточных инвестиций - построить сложные, крупномасштабные распределенные системы, которые позволяют полностью автоматизировать развертывание с нулевым временем простоя.

Я: есть такие практики как сине-зеленое развертывание, канареечное, теневое, использование фиче-флагов и т.д.

В основном большинство проблем развертывания вызвано сложным, хрупким процессом развертывания. Как правило, это является результатом трех факторов:

- ПО часто пишется без учета возможности дальнейшего развертывания. Общим симптомом здесь является то, что требуются сложные, организационные развертывания, поскольку ПО предполагает, что его среда и зависимости будут настроены под него совершенно особым образом, и не допускает никаких отклонений от этих ожиданий. *Я: одна ошибка и ты ошибся)* Это дает мало полезной информации админам о том, что идет не так и почему ПО работает неправильно. Эти характеристики также говорят о плохом проектировании для распределенных систем.
- Вероятность неудачного развертывания существенно возрастает, когда в процессе развертывания необходимо вручную внести изменения в производственную среду. Ручные изменения могут легко привести к ошибкам, вызванным вводом текста, ошибками копипасты или плохой или устаревшей документации. Кроме того, среды, конфигурация которых управляется вручную, часто существенно отличаются друг от друга (проблема известная как “дрейф конфигурации”), что приводит к значительным объемам работы во время развертывания, когда операторы отлаживают систему, чтобы понять различия в конфигурации, внося вручную дополнительные изменения, которые потенциально могут добавить еще проблем. *Я: знакомая ситуация, не правда ли;) ?*
- В рамках сложных развертываний часто требуется несколько итераций передачи полномочий между командами, особенно в разрозненных организациях, где админы баз данных - DBA, сетевые админы, безопасники, тестировщики - QA и разработчики работают в отдельных командах.

Чтобы уменьшить проблемы при развертывании, мы должны:

- Обеспечить создание систем, которые предназначены для легкого развертывания в нескольких средах, могут обнаруживать и допускать сбои в своих средах и независимо обновлять различные компоненты системы
- Обеспечить возможность автоматического воспроизведения состояния производственных систем (за исключением продуктивных данных) из информации в системе контроля версий.
  Я: В системе контроля версий важно сохранять не только код, но и конфигурации и само описание инфраструктуры из которого ее можно развернуть (этот подход называется IaC - Infrastructure as a Code)
- Встроить интеллектуальные решения в приложение и платформу, чтобы процесс развертывания был насколько возможно простым.

Выгорание

Выгорание - это физическое, умственное или эмоциональное истощение, вызванное переутомлением или стрессом, но это больше, чем просто переутомление или стресс. Выгорание приводит к тому, что вещи, которые мы когда-то любили в нашей работе и жизни, начинают казаться незначительными и скучными. Оно часто проявляется как чувство беспомощности и связано с патологическими культурами и непродуктивной, бесполезной работой.
Последствия выгорания огромны как для отдельных людей, так и для команд и их организаций. Стрессовая работа крайне вредна для физического здоровья.
Симптомы выгорания включают в себя чувство усталости, цинизма или неэффективности, отсутствие чувства выполненного долга и вообще негативно влияет на другие аспекты вашей жизни. В крайних случаях выгорание может привести к семейным проблемам, тяжелой клинической депрессии и даже к самоубийству.
Стресс на работе также влияет и на работодателей, ведет к текучести кадров. Таким образом, работодатели обязаны заботиться о своих сотрудниках и обеспечивать условия, в которых персонал не сгорал бы на работе.

Выгорание можно предотвратить или обратить вспять, и DevOps может в этом помочь. Организации могут устранить условия, которые приводят к выгоранию, создавая благоприятную рабочую среду, убеждая людей, что их работа имеет значение, и обеспечивая понимание сотрудниками того, как их собственная работа связана со стратегическими целями компании.

Руководители зачастую с благими намерениями пытаются исправить человека, игнорируя рабочую среду, хотя изменение окружающей среды гораздо более важно для долгосрочного успеха.
Руководители, которые хотят предотвратить выгорание сотрудников, должны сосредоточить свое внимание и усилия на следующих действиях:

- Создать уважительную, поддерживающую рабочую среду, в которой на неудачах учатся, а не используют их для обвинений.
  Я: Это называется blameless-culture - она работает в отношении многих, почти для всех, т.к люди зачастую делают ошибки непреднамеренно, например, по незнанию или из за плохо выстроенных процессов, которые допускают ситуации, приводящие к ошибкам. Однако, в отношении людей, делающих специально диверсии или неадекватные действия - blameless-culture, разумеется, не работает - и это должно наказываться.
- Транслировать сильное чувство цели.
- Спрашивать сотрудников, что мешает им достичь своих целей, а затем исправлять это.
- Предоставлять сотрудникам время, пространство и ресурсы для экспериментов и обучения.
- Сотрудникам должны быть предоставлены полномочия принимать решения, которые влияют на их работу и их рабочие места, особенно в тех областях, где они несут ответственность за результаты.

Общие проблемы, которые могут привести к выгоранию:

- Перегрузка работой: требования работы превышают человеческие пределы.
- Отсутствие контроля: неспособность влиять на решения, которые затрагивают вашу работу.
- Недостаточное вознаграждение: недостаточно финансовое, институциональное или социальное вознаграждение.
- Распад сообщества: неблагоприятная рабочая среда.
- Отсутствие справедливости: отсутствие справедливости в процессе принятия решений.

Исследование показало, что улучшение технических практик (например, тех, которые способствуют непрерывной доставке) и бережливых практик (например, в области бережливого менеджмента и бережливого управления продуктами) уменьшает чувство выгорания у сотрудников.

Главные факторы, которые сильнее всего коррелируют с высоким уровнем выгорания:

- **Организационная культура**
  Сильное чувство выгорания обнаруживается в организациях с патологической, ориентированной на власть культуру. Лидеры в конечном счете несут ответственность за создание благоприятной и уважительной рабочей среды, и они могут сделать это, создав среду без обвинений стремясь учиться на неудачах и создавая общее чувство цели.
- **Боль развертывания**
  Сложные, болезненные развертывания, которые должны выполняться в нерабочее время, способствуют высокому стрессу и ощущению отсутствия контроля ситуации. Проблемы, возникающие после развертывания, также важно отследить. Ломающиеся системы, которые постоянно вызывают ваш дежурный персонал в нерабочее время, разрушительны и нездоровы. При правильной практике развертывания не должны быть болезненными событиями. Руководители и лидеры должны спросить свои команды, насколько болезненно происходит их развертывание, и исправить то, что причиняет наибольшую боль.
- **Эффективность лидеров**
  Обязанности руководителя группы включают ограничение работы в процессе и устранение препятствий, чтобы команда могла выполнять свою работу. Чем эффективней руководитель, тем ниже уровень выгорания.
- **Организационные инвестиции в DevOps**
  Организации, которые инвестируют в развитие навыков и возможностей своих команд, получают лучшие результаты. Инвестиции в обучение и предоставление людям необходимой поддержки и ресурсов (включая время) для приобретения новых навыков имеют решающее значение для внедрения DevOps.
  Я: DevOps нельзя внедрить в одиночку, DevOps не существует в вакууме, DevOps - это про развитие культуры как технической, так и социальной. И чтобы двигать и развивать DevOps не достаточно простого согласия на словах, должна быть реальная поддержка руководства и выделение соответствующих ресурсов: времени, людей, инфраструктуры, ресурсов на исследования и изучения. Крайне сложно строить DevOps, например не имея

поддержки ресурсами со стороны эксплуатации. На следующие мысли меня натолкнула книга Team Topologies, причем в ряде крупных финтех организациях я встречался с представителями Enabling Teams (Отделы внедрения), описанных в данной книге.
Многие команды не могут выделить время на исследования, изучения новых практик, получение новых скиллов, т.к должны делать свою текущую работу (stream-aligned teams - SAT) Enabling team (ET) призвана восполнить это. Ее цель - помогать остальным командам. ET тратит время на изучение практик, фреймворков, инструментов, стека технологий и проводит различные исследования, учитывая контекст своей организации, с целью применить их. Полученные знания ET масштабирует на остальные SAT-команды (максимально распространяя знания), помогая им применять и внедрять новые практики и технологии, таким образом им не приходится тратить время, которого у них и так нет на изучение практик с нуля. ET носит глобальный характер в организации. Она имеет коллаборативную природу, ET стремится помогать другим подразделениям, командам организации, вникая в их проблемы (технического характера) и помогая с их решением. Таким образом ET представляет собой отдел технического консалтинга, но учитывающего специфику своей организации. ET не становится чем-то вроде "башни слоновой кости" и не позиционирует себя выше других SAT-команд, а наоборот помогает им и делится знаниями. ET как правило консультирует не единичных людей, а целые команды. Цель ET - увеличить автономность SAT-команд, фокусируясь в первую очередь на их проблемах, а не на технической части (т.е не внедрять технологию ради технологии, а подыскивать и приспосабливать необходимую, чтобы она решала проблемы). SAT-команды не зависят перманентно от ET, получив необходимую помощь и научившись новым практикам, совместно примененными с ET - они становятся более самостоятельными и технически подкованными. Т.е после помощи ET SAT-команда "растет как личность" становится умнее, мудрее и самостоятельней. ET помогает в технических проблемах, например, таких как: выстраивание CI/CD процессов, внедрение автоматизированного тестирования, Observability-системы, а также других DevOps-практик и т.д. ET-постоянно делится своими навыками с SAT-командами, ведет базу знаний, устраивает митапы, тренинги, занимается DevRel'ом. Таким образом ET-команда выстраивает единый стандарт и базу знаний, распространяя ее на всю организацию. Но наличие ET-команды не подразумевает, что лишь они занимаются развитием и изучением новых технологий, они просто помогают быстрее другим командам разобраться и помочь, они представляют собой внутренний консалтинг, плечо помощи. Развитие и изучение - общее дело на благо развития как личного, так и организации.

- **Организационная эффективность**
  Бережливое управление и методы непрерывной доставки помогают повысить эффективность доставки ПО, что в свою очередь повышает организационную эффективность. В основе бережливого менеджмента лежит предоставление сотрудникам необходимого времени и ресурсов для улучшения собственной среды, которая поддерживает эксперименты, трансформирует ошибки в обучение и позволяет сотрудникам принимать решения, влияющие на их работу. Это также означает создание пространства для сотрудников, позволяющего делать новую, творческую, добавляющую ценность работу. Например, в Google 20% рабочей недели отводится на эксперименты и самообразование. Более того, наличие этого фактора выравнивает организационные и личные цели сотрудников и следовательно сокращает выгорание.
  Я: Нельзя говорить, что организация современная и выпускает надежное ПО и в то же время агонизировать при ручном деплое сервисов. В книге приводится пример с деревьями: нельзя вырубать лес и говорить, что организация поддерживает экологию. Подобное различие ценностей с реальными действиями вызывает когнитивный диссонанс и ведет к выгоранию.

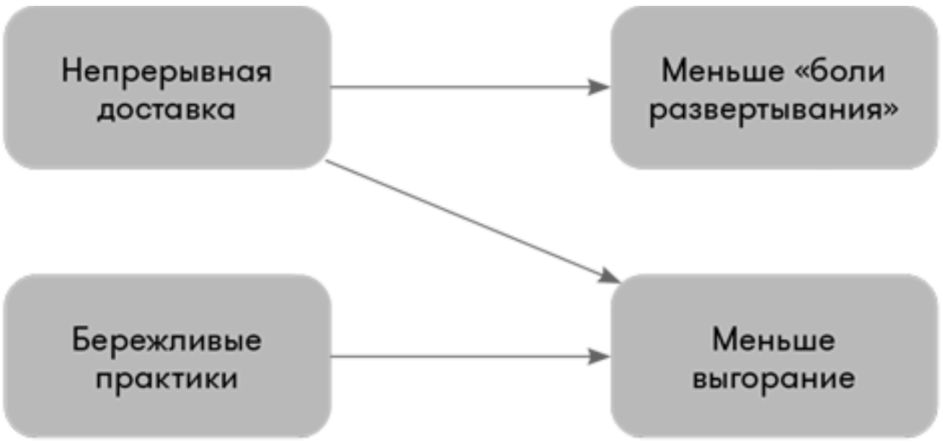


Рис. 9.1. **Влияние технических и бережливых практик на рабочую жизнь**

Глава 10 - Удовлетворенность сотрудников, их идентификация с организацией и вовлеченность

Люди находятся в центре каждой технологической трансформации. Под давлением рынка, вынуждающего как можно быстрее поставлять новые технологии и решения, важность привлечения, удержания и вовлечения персонала больше, чем когда бы то ни было.

Чтобы понять вовлеченность сотрудников в контексте технологических преобразований и DevOps, авторы рассмотрели ее через призму широко используемого стандарта лояльности - Net Promoter Score(NPS).
В высокоэффективных компаниях лояльность сотрудников, измеряемая индексом чистой лояльности (eNPS), выше.
NPS рассчитывается на основе одного простого вопроса: насколько вероятно, что вы порекомендуете нашу компанию/продукт/услугу другу или коллеге?

NPS оценивается по шкале от 0 до 10 и классифицируется так:

- Клиенты, которые дают оценку 9 или 10, считаются “промоутерами”. Промоутеры создают наибольшую ценность для компании, потому что они, как правило, покупают больше, остаются дольше, генерируют позитивное “сарафанное радио”, а приобрести и удержать их стоит меньше.
- Те, кто дает оценку 7 или 8, считаются “пассивами”. Пассивы - это удовлетворенные, но гораздо менее восторженные клиенты. Они меньше склонны представлять рекомендации и с большей вероятностью переметнутся к конкурентам, если появится что-то лучше.
- Те, кто дает оценку от 0 до 6, считаются “недоброжелателями”. Их дороже приобретать и удерживать, они быстрее “дезертируют” и могут повредить бизнесу через негативное “сарафанное радио”.

NPS = процент промоутеров - процент недоброжелателей

Чтобы оценить лояльность сотрудников - eNPS, в исследовании задавались два вопроса:

- Вы бы порекомендовали свою ОРГАНИЗАЦИЮ в качестве места работы другу или коллеге?
- Вы бы порекомендовали свою КОМАНДУ в качестве места работы другу или коллеге?

Исследования показали, что сотрудники высокоэффективных организаций в 2,2 раза чаще рекомендуют свою компанию как отличное место работы и в 1,8 раза чаще порекомендовали бы другу свою команду и еще это коррелирует с лучшими результатами бизнеса (увеличение доходов до 2,5 раз по сравнению с компаниями с низким уровнем вовлеченности).

Индекс чистой лояльности сотрудников коррелирует со следующими конструкциями:

- степень, в которой организация собирает отзывы клиентов и использует их для информирования команд разработки продуктов и функций.
- способность команд визуализировать и понимать поток продуктов или функций через разработку и далее на протяжении всего пути до клиента.
- степень, в которой сотрудники идентифицируют себя с ценностями и целями своей организации, и усилия, которые они готовы приложить, чтобы сделать организацию успешной.

Когда сотрудники видят связь между выполняемой ими работой и ее позитивным воздействием на клиентов, они сильнее идентифицируют себя с целями компании, что приводит к улучшению доставки ПО и организационной эффективности.

Люди - это самый главный актив любой организации, но, к сожалению, часто к ним относятся как к расходным материалам. Когда лидеры инвестируют в своих сотрудников и позволяют им делать свою работу лучше, сотрудники сильнее идентифицируют себя с организацией и готовы приложить дополнительные усилия, чтобы помочь ей добиться успеха. В свою очередь, организации получают более высокий уровень эффективности и производительности, что приводит к лучшим результатам для бизнеса.

Рис. 10.1. **Влияние технических и бережливых практик на идентификацию**

Чтобы измерить уровень идентификации сотрудника с организацией, в исследовании спрашивалось в какой степени (по шкале Ликерта, описание ниже) они согласны с следующими утверждениями:

- Я рад, что выбрал работу в этой компании, а не в другой.
- Я рассказываю об этой компании друзьям как об отличном месте работы.
- Я готов приложить гораздо больше усилий, чем обычно ожидается, чтобы помочь моей компании добиться успеха.
- Я считаю, что мои ценности и ценности компании очень схожи.
- В целом люди, занятые в моей компании, работают в направлении одной цели.
- Я чувствую, что моя компания заботится обо мне.

Шкала Ликерта:

1. Полностью не согласен
2. Не согласен
3. Где-то посередине
4. Согласен
5. Полностью согласен.

Для измерения идентификации в ваших собственных командах вы можете усреднить оценку по 6 пунктам в одну оценку для идентификации человека.

Чем больше сотрудник идентифицирует себя с компанией - тем лучше и компании и сотруднику. Компании - прибыль и эффективность, сотруднику - радость и удовлетворение от работы, заинтересованность и отсутствие выгорания. Это контрастирует с тем, как многие компании все еще работают: требования спускаются командам разработчиков, которые затем должны доставлять большие части работы в пакетном режиме. В этой модели сотрудники ощущают слабый контроль над продуктами и опытом клиентов, которые они создают, а также слабую связь с организациями в которых они работают. Это чрезвычайно демотивирует команды и приводит к тому, что сотрудники чувствуют себя эмоционально оторванными от своей работы, и, соответственно, к худшим результатам.
Важно ценить своих сотрудников. Однажды лидер одной из компаний из списка Fortune 500 спросил Эйдриана Кокрофта, главного архитектора облака Netflix, откуда он берет своих удивительных людей. На что Кокрофт ответил: “Я нанял их у вас!”

Рис. 10.2. **Влияние технических и бережливых практик на удовлетворенность работой**

Этот замкнутый круг непрерывного совершенствования и обучения является отличительной чертой успешных компаний, позволяющий им внедрять инновации, опережать конкурентов - и побеждать.

Как DevOps способствует удовлетворенности работой?

Хотя DevOps - это прежде всего, культура, важно отметить, что удовлетворенность работой сильно зависит от правильных инструментов и ресурсов для ее выполнения. Инструменты являются важным компонентом практики DevOps, и многие из этих инструментов подразумевают автоматизацию. Кроме того, хорошая техническая практика предопределяет удовлетворенность работой. Рассматривая показатели, которые сильно коррелируют с удовлетворенностью работой, мы видим некоторые общие черты. Такие методы, как проактивный мониторинг и автоматизация тестирования и развертывания, автоматизируют служебные задачи и требуют от людей принятия решений на основе цикла обратной связи. Вместо того, чтобы управлять задачами и увязнуть в рутине, люди получают возможность принимать решения, используя свои навыки, опыт и мнение.

Глава 11 - Лидеры и руководители

Лидерство оказывает мощное влияние на результаты.

Быть лидером не означает, что у вас есть люди, отчитывающиеся перед вами на организационных диаграммах. Лидерство заключается в том, чтобы вдохновлять и мотивировать тех, кто вас окружает. Хороший лидер влияет на способность команды доставлять код, создавать хорошие системы и применять принципы бережливого производства к тому, как команда управляет своей работой и разрабатывает продукты.

Трансформационное лидерство означает, что лидеры вдохновляют и мотивируют своих последователей на достижение более высоких результатов, апеллируя к их ценностям и чувству цели, способствуя тем самым широкомасштабным организационным изменениям. Такие лидеры побуждают свои команды работать над достижением общей цели через свое видение,ценности, общение, собственный пример и очевидную заботу о личных потребностях своих последователей. Есть сходство между “служебным” лидерством и трансформационным лидерством, но они отличаются фокусом внимания лидера. “Служебные” лидеры сосредотачиваются на развитии и эффективности своих последователей, в то время как трансформационные лидеры сфокусированы на том, чтобы добиться от своих последователей идентификации себя с организацией и участия в поддержке организационных целей.

Трансформационное лидерство имеет большое значение для:

- установления и поддержки созидательных и высоконадежных культурных норм
- создания технологий, обеспечивающих производительность разработчиков, сокращение времени развертывания кода и поддержку более надежных инфраструктур
- поддержания экспериментов и инноваций в команде, а также более быстрого создания и внедрения улучшенных продуктов
- работы над преодолением организационной разобщенности для достижения стратегического выравнивания ценностей

Однако, бывают и такие порочные консервативные лидеры (ретрограды), которые мешают командам вносить изменения, необходимые для улучшения доставки ПО и организационной эффективности.
Лидеры должны обладать полномочиями и бюджетом для проведения крупномасштабных изменений, которые часто необходимы для обеспечения “прикрытия с воздуха и поддержки” в ходе преобразований и для изменения стимулов целых групп технических специалистов - будь то в области разработки, контроля качества, техподдержки или информационной безопасности. Лидеры - это те, кто задает тон организации и укрепляет желаемые культурные нормы.
Чтобы измерить трансформационное лидерство в исследовании использовались следующие вопросы из работы Рафферти и Гриффина.

Мой лидер или руководитель:

- **видение:**
  - имеет четкое понимание, куда мы идем
  - имеет четкое представление о том, где он хочет, чтобы наша команда была через 5 лет
  - имеет четкое представление о том, куда идет организация
- **вдохновляющее общение:**
  - говорит вещи, которые заставляют сотрудников гордиться тем, что они являются частью этой организации
  - позитивно отзывается о работе подразделения
  - поощряет людей видеть изменяющуюся среду как ситуацию, полную возможностей
- **интеллектуальная стимуляция:**
  - заставляет меня думать о старых проблемах по-новому
  - транслирует идеи, которые заставили меня переосмыслить некоторые вещи, в которых я никогда ранее не сомневался
  - заставляет меня переосмыслить некоторые из моих основных предположений о моей работе
- **поддерживающее лидерство:**
  - учитывает мои личные чувства, прежде чем действовать
  - ведет себя таким образом, чтобы заботиться о моих личных потребностях
  - видит, что интересы сотрудников учитываются должным образом
- **личное признание:**
  - поощряет, когда я делаю работу лучше, чем на среднем уровне
  - признает улучшение качества моей работы
  - хвалит лично меня, когда я делаю выдающуюся работу

Исследование показало, что эти характеристики сильно коррелируют с эффективностью доставки ПО.
Высокоэффективные команды сообщили, что у них есть лидеры с самыми высокими показателями во всех измерениях: видение, вдохновляющая коммуникация, интеллектуальная стимуляция, поддерживающее лидерство и личное признание.
Напротив, низкоэффективные команды сообщили о самых низких уровнях эти лидерских характеристик.
Команды с наименее трансформирующими лидерами с гораздо меньшей вероятностью достигают высоких результатов.
Хотя мы часто слышим об успехах DevOps и технологических преобразований, исходящих от рядовых сотрудников, гораздо легче добиться успеха, когда у вас есть поддержка руководства.
Исследования также показали, что трансформационное лидерство в значительной степени коррелирует с eNPS. Мы видим трансформационных лидеров там, где сотрудники счастливы, лояльны и вовлечены.
Лидеры в одиночку не могут достичь высоких результатов DevOps. Они нуждаются в том, чтобы их команды выполняли работу на подходящей архитектуре, с

использованием хороших технических практик и принципов бережливого производства, а также всеми другими факторами.

Рис. 11.1. **Влияние трансформационного лидерства на технические возможности и возможности бережливого производства**

Трансформационное лидерство:

- помогает создавать отличные команды
- позволяет командам перестраивать свои системы и внедрять необходимые методы непрерывной доставки и бережливого управления
- коррелирует с высокой эффективностью
- поддерживает эффективное общение и сотрудничество между членами команды в достижении организационных целей

Роль руководителей

Лидеры играют решающую роль в любом технологическом преобразовании. Когда эти лидеры являются руководителями, они могут еще сильнее влиять на изменения.

Руководители:

- Несут ответственность за людей, а часто и за бюджеты и ресурсы в организациях. В лучшем случае руководители также являются лидерами и принимают на себя характеристики трансформационного лидерства.
- Играют важную роль в соединении стратегических целей бизнеса с работой своих команд.
- Играют важную роль в соединении стратегических целей бизнеса с работой своих команд. Могут многое сделать для повышения эффективности своей команды, создавая рабочую среду, в которой сотрудники чувствуют себя в безопасности, инвестируя в развитие возможностей своих сотрудников и устраняя препятствия в их работе.
- Могут улучшить ситуацию, включив конкретные методы DevOps в свои команды и заметно инвестируя в DevOps и в профессиональное развитие своих сотрудников.

- Могут способствовать значительному улучшению эффективности доставки ПО, принимая меры, чтобы сделать развертывание менее болезненным.
- Должны сделать показатели эффективности прозрачными и приложить усилия, чтобы привести их в соответствие с целями организации, а также делегировать больше полномочий своим сотрудникам. Знание - сила, и вы должны дать силу тем, кто обладает знанием.
- Улучшают культуру

Исследования показали, что инвестиции в DevOps сильно коррелируют с эффективностью доставки ПО.

Инвестиции в инициативы DevOps:

- Обеспечьте доступность существующих ресурсов для всех сотрудников организации. Создайте пространство и возможности для обучения и совершенствования.
- Создайте специальный бюджет на обучение и убедитесь, что люди знают о нем. Кроме того, дайте вашим сотрудникам свободу выбора обучения, которое их интересует. Этот бюджет на обучение может включать в себя выделенное в течение дня время для использования тех ресурсов, которые уже существуют в организации.
- Поощряйте сотрудников участвовать в технических конференциях по крайне мере один раз в год и делиться со всей командой тем, что они узнали.
- Введите внутренние хакатоны, когда кроссфункциональные команды могут собраться вместе, чтобы работать над проектом.
- Поощряйте команды к проведению дней для погашения тех.долга
- Регулярно проводите внутренние мини-конференции DevOps.
- Предоставьте сотрудникам специальное время для экспериментов с новыми инструментами и технологиями. Выделите бюджет и инфраструктуру для специальных проектов.

Советы по улучшению культуры, поддержке ваших команд и кроссфункциональному сотрудничеству:

- **Выстраивайте доверительные отношения с вашими коллегами в других командах.**
  Построение доверия между командами - это самое важное, что вы можете сделать. Доверие строится на сдержанных обещаниях, открытом общении и предсказуемом поведении даже в стрессовых ситуациях. Ваши команды смогут работать более эффективно, а эти отношения будут сигнализировать организации, что кроссфункциональное сотрудничество ценится.
- **Поощряйте специалистов к перемещению между отделами.**
  Админы или другие инженеры могут обнаружить, что на позиции в другом отделе они приобрели интересные им навыки. Такое горизонтальное перемещение может быть невероятно ценным для обеих команд. Практикующие специалисты приносят ценную информацию о процессах и проблемах в свою

новую команду, а члены их предыдущей команды получают в лице этих людей естественную точку входа, когда дело доходит до сотрудничества.
Я: Если не давать возможности ротации сотрудникам, то сотрудник может либо выгореть, либо просто уйти в другую компанию. Причем, уход может быть резким и создаст вакуум в команде, а организации придется потратиться на поиск и внедрение нового сотрудника в команду, что занимает приличное время и может создать стрессовую ситуацию в самой команде, связанную с переработками и онбордингом нового сотрудника. В случае ротации - этот переход был плановым и позволил бы избежать подобных проблем.

- **Активно ищите, поощряйте и вознаграждайте работу, которая облегчает сотрудничество**
  Убедитесь, что успех поддается повторению, и обратите внимание на факторы, облегчающие сотрудничество.
- **Устраивайте тренировочные игровые дни по тестированию восстановления в случае аварийных ситуаций**
  Многие крупные технологические компании проводят тесты аварийного восстановления или “игровые дни”, в которых сбои моделируются или фактически создаются в соответствии с заранее подготовленным планом и команды должны работать вместе для поддержания или восстановления необходимого уровня обслуживания.
- **Создайте бюджет на обучение и продвигайте его внутри команды**
  Подчеркните, насколько организация ценит климат обучения, вкладывая ресурсы в возможности формального образования.
- **Обеспечьте вашей команде ресурсы для участия в неформальном обучении и пространство для изучения идей**
  Выделяйте часть времени на самообразование сотрудников.
- **Сделайте ошибки безопасными**
  Если неудача наказуема, люди не будут пробовать новые вещи. Рассмотрение неудач, как возможностей для обучения и исследование ошибок для выработки способов улучшения процессов и систем без необходимости обвинений помогают людям чувствовать себя комфортно, принимая риски (разумные), и способствуют созданию культуры инноваций.
- **Создайте возможности и пространства для обмена информацией**
  Установите регулярные возможности для сотрудников делиться своими знаниями.
- **Поощряйте обмен и инновации, устраивая демонстрационные дни и митапы**
  Нетворкинг и обмен знаниями

Эффективно используйте инструменты:

- **Убедитесь, что ваша команда сама может выбирать инструменты**
  Если нет веской причины не делать этого, инженеры должны выбирать подходящие инструменты на свой выбор. Это очень влияет на удовлетворенность работой и повышает эффективность доставки ПО. Если ваша организация должна стандартизировать инструменты, убедитесь, что закупки и финансы действуют в интересах команд, а не наоборот.
- **Сделайте мониторинг приоритетом**
  Усовершенствуйте свою инфраструктуру и систему мониторинга приложений, а также убедитесь, что вы собираете информацию о нужных сервисах и используете эту информацию с пользой. Наглядность и прозрачность, которые обеспечиваются эффективным мониторингом, бесценны. Проактивный мониторинг тесно связан с эффективностью и удовлетворенностью работой, а это является ключевой частью прочной технической основы.

Хотя многие истории успеха DevOps подчеркивают фантастические усилия вовлеченных технических команд, опыт и исследования авторов показывают, что технологические преобразования выигрывают от действительно вовлеченных трансформационных лидеров, которые способны поддерживать и усиливать работу своих команд. Эта поддержка обеспечивает ценность для бизнеса, поэтому организация м было бы разумно рассматривать развитие лидерства как инвестиции в свои команды, свои технологии и свои продукты.