Хаос-испытания отказоустойчивости информационных систем

Contents

Зачем этот курс?

Мы учим моделировать аномальные ситуации в работе дорогих и сложных ИТ-систем, чтобы вы находили и устраняли отказы и сбои до того, как они произошли.

Внедряя практику хаос-испытаний на уровне компании, вы сможете перейти от режима реагирования на инциденты к их проактивному предотвращению.

Отказоустойчивость

Микросервисные и распределенные архитектуры позволяют создавать более сложные и масштабируемые ИТ-системы. В эксплуатационной среде неизбежны сбои, к которым одни системы готовы, а другие — нет. Проверку отказоустойчивости обеспечивает новый вид тестирования, который называется chaos engineering.

От Netflix до CNCF

Впервые chaos engineering стали применять в компании Netflix в ходе миграции видеосервиса на облачную инфраструктуру. В 2011 году Netflix выпустил в открытый доступ первый инструмент для проведения хаос-испытаний — Chaos Monkey.

Лидеры направления chaos engineering в настоящее время это Amazon, Bloomberg, Netflix и Alibaba, крупнейшие провайдеры SaaS-сервисов для хаос-испытаний — американская Gremlin и европейская ChaosIQ.

С 2017 года инструменты chaos engineering стали официальной частью ландшафта технологий, которые Cloud Native Computing Foundation (CNCF) рекомендует для Kubernetes.

Хаос на российском рынке (по количеству вакансий)

На сервисе Headhunter сейчас более 2000 вакансии по DevOps, и более 200 по SRE и всего 5-6 по chaos engineering.

Область	08/2022	03/2023
DevOps	1942	2243
SRE	192	204
Chaos engineering	6	5

Хаос-испытаниями занимается достаточно небольшой круг продвинутых команд и компаний, которые чаще всего работают с массовыми, высоконагруженными сервисами.

Кому подходит и кому не рекомендуется chaos engineering мы рассказываем здесь.

Перейти к рекомендациям

Где применяется chaos engineering?

Для каких компаний подходит chaos engineering? На наш взгляд, chaos engineering наиболее полезен компаниям, которые:

- работают в сферах, где потребитель чувствителен к уровню надежности ИТ-сервисов;
- могут показать ценность надежной работы сервисов акционерам компании;
- ожидают роста требований по надежности со стороны регуляторов (например, в финансовом секторе);
- активно используют микросервисы и распределенные системы;
- имеют достаточные компетенции в проектировании отказоустойчивого ПО;
- планируют миграции на новые архитектуры или стек технологий, при этом хотят убедиться в отказоустойчивости нового решения;
- намерены совершенствовать техническую и организационную культуру компании.

Воспользоваться chaos engineering будет сложнее компаниям, у которых:

- продукт или сервис находятся в начальной стадии разработки или прототипирования, архитектура приложений будет меняться;
- нет собственной разработки, ведется только настройка и эксплуатация приобретенного ПО;
- не стоит задача создания отказоустойчивого ПО;
- не предполагается масштабирование сервиса на большое число пользователей;
- по архитектуре преобладают монолитные приложения;
- специализация преимущественно на front-end разработке;
- не документируются требования или SLA используемых систем;
- не выстроен современный релизный цикл ПО или релизы занимают длительное время;
- проводится мало других видов тестирования помимо хаос-инжиниринга;
- нет мониторинга надежности сервисов в стадии эксплуатации;
- не ведется история и разбор инцидентов в области надежности;
- надежность сервиса не критична для пользователей;
- эксплуатируются legacy и end-of-life приложения, не подлежащие рефакторингу или доработке.

Уровни зрелости chaos engineering в компании

Внедрение хаос-испытаний в компании - многошаговый процесс, который обычно начинается с разовых экспериментов, показывающих нештатное поведение конкретной системы в случае простых атак. Такие атаки обычно воспроизводят инциденты, которые уже происходили в ходе эксплуатации системы.

По мере накопления опыта проведения испытаний расширяется количество тестируемых атак и набор инструментов для их внедрения, используются средства для автоматизации хаос-экспериментов, улучшается предоставление необходимой инфраструктуры (стендов) и вырабатывается собственная методика проведения хаос-испытаний. Команды лучше понимают поведение сложных распределенных систем в условиях стресса.

Подготовительные этапы обеспечивают переход к разработке систем, способных к самовосстановлению при возникновении неблагоприятных условий в эксплуатационной среде. Для этого может быть создан профильный центр компетенций, лучшая практика разработки таких систем распространяется на всю компанию (через выделенные роли в командах, встраивание в релизный цикл и SLA систем) и подтверждается хаос-испытаниями, охватывающими большинство систем в компании.

На продвинутых этапах компания распространяет методологию внутри экосистемы и на партнеров. Возможные этапы внедрения хаос-испытаний в крупной компании показаны на рисунках ниже.

Уровни зрелости внедрения хаос-испытаний (для крупной компании)

3 и 4 уровень зрелости являются критическими для перехода к отказоустойчивым системам в масштабе компании

Стадия	Количество систем (охват)	Отношение команд	Поведение тестируемой системы	Идея	Кто занимается
1	1-5	Blocker	Неопределенное поведение	Понять отдельные сбои	Энтузиасты-одиночки
2	10-15	Blocker		Понять более сложные сбои	Продвинутые команды
3	15-30	Blocker/Enabler		Озадачить разработчиков	
4	>30	Enabler	Система падает ожидаемым образом	Выявить лучшие практики разработки	
5	Все внутри компании	Enabler	Самовосстановление системы = обязательное требование	Получить выгоду от лучших практик	Центр компетенций
6	Экосистема	Enabler	Отказоустойчивость цепи систем	Улушить интеграции	
7	Смежные системы, архитектурный стандарт	Enabler	Отказоустойчивость по дефолту	Не думать о том, какая практика лучшая	Присоединяются партнеры

Blocker = хаос-испытания тормозят работу команд, команды избегают дополнительных испытаний системы.
Enabler = команды хотят проводить хаос-испытания, чтобы подтвердить качество и технический уровень внедряемых систем и сервисов.

Стадия	Методология	Инструменты и виды атак	Автоматизация тестов	Инфраструктура для тестирования	Стоимость испытаний
1	По книжкам и мануалам	1-2 инструмента внедрения атак, простые атаки	Нет автоматизации, ручное тестирование	Не ясно, где взять и как настроить	Не рассчитывается
2	Предложения по собственной методике			Ясно, какая инфраструктура нужна, но собирать ее долго (от нескольких дней)	Высокая
3	Своя базовая методика	3-5 инструментов внедрения атак, собственная матрица атак	Малая автоматизация	Есть типовое решение по настройкам	
4	Готовность к тиражированию		Средняя автоматизация	Стенд для тестирования настраивается за часы	Начинает снижаться
5	Обобщение и пересмотр опыта / своя методология	Сложные сценарии испытаний, повторные испытания (регресс), система хранения отчетов и собственные инструменты	Полная автоматизация, которая представлена отдельным комплексным продуктом.	Стенд адаптирован под цели испытаний	Резко сокращается благодаря автоматизации и масштабу
6	Адаптация внутри экосистемы				
7	Методология влияет на рынок	Предложить свои инструменты на рынок			

Начиная с 4 этапа компания также может быть заинтересована в применении средств машинного обучения и искусственного интеллекта для обработки данных мониторинга и хаос-испытаний (AIOps), а также совместном тестировании надежности и безопасности систем (DevSecOps).

Содержание и ожидаемые результаты курса

С помощью курса вы получаете доступ к современной практике проведения хаос-испытаний. Курс позволяет сократить время на подготовку первого хаос-испытания и ускорить распространение практики chaos engineering на уровне компании.

Слушатели курса осваивают методологию хаос-испытаний и в рамках практикумов по Linux и Windows учатся настраивать и запускать инструменты для внедрения атак.

Основные части курса

1. Что такое хаос-испытание? Понятие аномалии. Гипотеза испытания.
2. Типовая схема и рекомендации по подготовке и проведению испытаний.
3. Классификация атак по видам систем.
4. Обзор инструментария для проведения атак.
5. Практикум “Инструменты внедрения атак для Linux”.
6. Практикум “Инструменты внедрения атак для Windows”.
7. Результаты хаос-испытания и дальнейшая работа с ними.

Курс проводится в течение одного дня, время занятий составляет до 6 часов в зависимости от размера группы и детализации практикумов по инструментам.

Выигрыши

По нашей оценке, курс экономит до 400 часов, которые необходимы для выбора и изучения документации по инструментам, а также на разработку и апробацию регламентов проведения хаос-испытаний.

Целевая аудитория

Курс будет полезен компаниям, которые:

- видят успешные примеры применения хаос-тестирования в своей отрасли;
- начали хаос-испытания в одной команде или проекте и хотят масштабироваться;
- планируют крупную миграцию архитектуры или смену технологического стека.

Возможные слушатели курса внутри компании:

- будущие и действующие хаос-инженеры;
- команды эксплуатации, DevOps, SRE;
- тестировщики и команды тестирования;
- разработчики;
- архитекторы;
- руководители технических подразделений и CTO.

Помимо этого курс будет полезен сотрудникам и руководителям подразделений, отвечающих за совершенствование технических и бизнес-процессов, технический аудит, анализ и управление рисками в сфере информационных технологий.

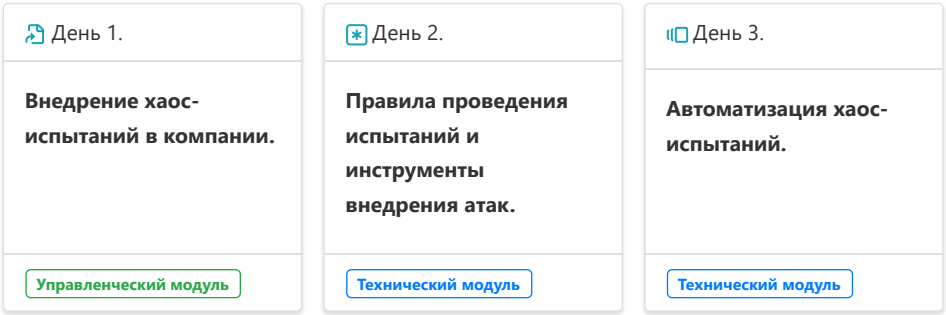
Адаптация курса

Возможна адаптация курса под индивидуальные запросы компании, конкретную аудиторию и уровень подготовки слушателей.

Темы будущих семинаров

- Инструменты внедрения атак под Kubernetes.
- Практикум “Проведение хаос-испытания гибридной системы.”
- Автоматизация хаос-испытаний.
- Оценка результативности (ROI) хаос-испытаний.
- Компетенции хаос-инженера и конфигурация команд.

Пример трехдневного курса



Авторы курса и контакты

Дмитрий Якубовский

Исполнительный директор, лидер направления Chaos Engineering в розничном блоке Сбербанка. Опыт работы в сфере Chaos Engineering с 2015 года.

Евгений Погребняк

Декан факультета финансовой экономики МГИМО, руководитель магистерской программы “Экономика ИТ и управление данными”.

Напишите нам в Telegram

1. Задайте интересующий вас вопрос по chaos engineering.
2. Расскажите о потребностях вашей компании в хаос-испытаниях.
3. Уточните содержание курса и закажите его проведение.

Telegram Дмитрий Якубовский (@yakubovskydn) Telegram Евгений Погребняк (@epoepo)

Глоссарий

Chaos engineering

Дисциплина и набор практик по проведению экспериментов, подтверждающих способность информационных систем противостоять неблагоприятным условиям в эксплуатационной среде (адаптировано из Principles of Chaos).

CNCF

Cloud Native Computing Foundation.

DevOps

Development and operations. Набор методик и инструментов, которые позволяют автоматизировать и интегрировать между собой процессы разработки и эксплуатации ПО.

K8s

Kubernetes.

Principles of Chaos

Манифест методологии chaos engineering, опубликован по адресу principlesofchaos.org.

SLA

Service Layer Agreement. Соглашение о предоставлении услуг – договор между заказчиком и поставщиком, описывающий согласованный уровень качества предоставления ИТ-услуги.

SRE

Site Reliability Engineering. Набор принципов, индикаторов и практик для обеспечения надежности ИТ-систем, который разработан и популяризируется компанией Google.

Атака

Создание неблагоприятных условий работы системы в ходе хаос-испытания, эмуляция отказа.

Гипотеза

Предположение о поведении системы при конкретном виде воздействия (атаке) на конкретный архитектурный элемент системы.

Инструмент

Утилита для проведения атаки.

Испытание

Контролируемое внесение изменений в условия работы сервиса или системы на тестовом стенде или в эксплуатационной среде под рабочей нагрузкой. Испытание проводится для подтверждения или опровержения заранее сформулированной гипотезы относительно поведения системы в неблагоприятных условиях.

Синонимы: хаос-испытание, эксперимент, хаос-эксперимент.

ПО

Программное обеспечение.

Система

Конкретная информационная система, которую мы изучаем или тестируем.

Синонимы: автоматизированная система (АС), сервис.

Точка отказа

Ситуация нештатного поведения системы, которая найдена в ходе хаос-испытания.

Версия 0.4.14

Статьи, книги, видео

Видео: выступления Дмитрия Якубовского

- Максим Козлов, Дмитрий Якубовский. Непростые истории Chaos Engineering. CodeFest Russia - 27 сент. 2022 г. - 46:50
- Дмитрий Якубовский, Максим Козлов. Путь в хаос или как мы строили Chaos Engineering в банке. Конференция ArchDays - 16 дек. 2019 г. - 43:24

Статья: Хаос-инжиниринг помогает DevOps'ам справляться со сложностями (GitHub)

Оригинал статьи: https://github.com/readme/featured/chaos-engineering
Дата: 10 мая 2022 года
Автор: Klint Finley
Перевод: Роман Татевосян, Евгений Погребняк

Телефон зазвонил, когда близился к концу последний час вашего on-call дежурства. Пользователи жалуются на то, что не могут отправлять сообщения, а это главная функция вашей платформы. Вы открываете программу и тут же видите проблему: кнопка «Отправить» не работает. Немного покопавшись, вы понимаете, что причиной проблемы является неисправность кэширующего сервера. Программа не может извлечь файл изображения с кнопкой, и на этом месте ничего не появляется. Вы перезапускаете кэширующий сервер. На следующий день, во время совещания по предотвращению инцидентов, frontend-команда добавляет запасную текстовую версию для всех элементов пользовательских интерфейсов на тот случай, если что-нибудь подобное повторится.

Проблемы с функционированием сервисов возникают все чаще и чаще, поскольку приложения развиваются и становятся сложнее. Продукты, которые выглядят для пользователей как нечто однообразное, в реальности представляют собой сложные комбинации различных элементов: серверов баз данных, API, микросервисов и библиотек – каждый из которых может выйти из строя в любой момент. Чтобы создать более устойчивые приложения и решить проблемы до того момента, когда они станут причиной сбоя, как это произошло в примере выше, DevOps-команды все чаще обращаются к дисциплине, которую называют «хаос-инжиниринг».

Самый простой пример: с помощью хаос-инжиниринга можно случайно выключить разные сервисы, чтобы посмотреть, как отреагирует система в целом. «Это помогает вам найти в системе ошибки, о существовании которых вы не знали», – говорит Нора Джонс, основательница и главный исполнительный директор стартапа Jeli – компании, которая специализируется на анализе инцидентов. Кроме того, она является соавтором книги «Хаос-инжиниринг: устойчивость системы на практике» и бывшим инженером по обеспечению устойчивости в компании Netflix.

Например, в 2018, до запуска интерактивного фильма «Черное зеркало: Брандашмыг», команда Netflix обнаружила, что небольшое хранилище ключей-значений оказалось довольно важным, хотя они считали его некритичным. «Это было чем-то вроде скрытой зависимости», – говорит инженер по обеспечению устойчивости в Netflix Алеш Плшек. - «Эта находка позволила нам сфокусироваться на устранении уязвимости, о которой мы раньше не знали. Хаос-инжиниринг действительно помогает нам быть уверенными в том, что мы инвестируем ресурсы в правильном направлении».

Интерес к дисциплине хаос-инжиниринга возрос с тех пор, как Netflix опубликовал исходный код собственного инструмента проведения хаос-экспериментов. Сегодня этот интерес связан с развитием таких проектов как Litmus, Chaos Toolkit и PowerfulSeal, которые предоставляют надежные и кастомизируемые инструменты для проведения хаос-экспериментов.

«Все организации, которые мне известны, сегодня в той или иной форме проводят хаос-эксперименты со своими системами», – говорит Джон Оллспоу, основатель Adaptive Capacity Labs и бывший технический директор Etsy. – «Хаос-эксперимент – одна из важнейших методик в области тестирования ИТ-продуктов. Методика хаос-эксперимента используется командами инженеров для повышения своей уверенности в работе сервиса в период создания программного обеспечения и, что даже важнее, при его модификации».

«Привлекательность хаос-инжиниринга заметна невооруженным глазом: во-первых, методика носит яркое название, а во-вторых, вам дают возможность крушить все подряд. Но хаос-инжиниринг – это не просто сеяние хаоса. Вы пытаетесь понять, как ваша система ведет себя», – говорит Сильван Эгуаш, ведущий разработчик проекта Chaos Toolkit и сооснователь стартапа Reliably. – «Это научный метод. Вы выдвигаете гипотезу – обычно что-то в духе “Система продолжит работать в нормальном режиме, даже если потеряет доступ к одному экземпляру базы данных”, – а затем вы проверяете эту гипотезу и смотрите на результаты».

Самое главное – хаос-инжиниринг представляет особый тип инженерного мышления, который помогает DevOps-командам справляться с неопределенностью. Благодаря инструментам с открытым исходным кодом хаос-инжиниринг находит свое применение на всех стадиях жизненного цикла программного обеспечения и помогает разработчикам переосмыслить то, как они создают, тестируют и поддерживают программные продукты.

«Успешные организации не двигаются в сторону упрощения – они становятся только сложнее», – говорит Оллспоу. – «Все больше инженеров признают, что не могут самостоятельно спрогнозировать будущее поведение, а иногда даже и понять текущее состояние своей ИТ-инфраструктуры. Методика хаос-инжиниринга является в некотором смысле движущей силой, которая воздействует на образ мыслей ИТ-сообщества и адаптирует его к потребностям нового мира».

Дисциплина хаоса

«Netflix начал работу над созданием хаос-инжиниринговой платформы в 2010 году, когда переводил свой сервис из физических дата-центров в облако», – поясняет Плшек. – «Они переживали за то, что может произойти в случае, если какой-либо инстанс AWS рухнет, или станут недоступны определенные сервисы. Экосистема наших ИТ сервисов меняется ежедневно, и в моменте по сути никто не знает, как это все устроено».

Netflix разработал сервис Chaos Monkey, исходя из простой идеи: отключать в произвольном порядке работу той или иной системы. «С началом применения хаос-экспериментов мы стали лучше осведомлены о потенциальных угрозах работы сервисов», – говорит Плшек.

Стриминговый сервис впервые рассказал о существовании Chaos Monkey в своем блоге, вызвав большой резонанс в ИТ-сообществе. В то время многие компании также переносили свои сервисы в облако или, по крайней мере, уже использовали API удаленных сервисов и микросервисные архитектуры. «Помню, как читал этот пост много лет назад, еще до того, как я присоединился к Netflix, и он полностью изменил мое представление о том, как должны развиваться сервисы», – говорит Плшек.

Впоследствии в 2012 году Netflix опубликовал исходный код Chaos Monkey. Это значительно облегчило работу других компаний по созданию собственных решений для хаос-экспериментов. «Chaos Monkey сильно повлиял на будущие платформы хаос-тестирования», – говорит Эгуаш.

По мере распространения идеи хаос-инжиниринга open source сообщество разработало больше возможностей для адаптации Chaos Monkey в разных организациях. «Мы пытались продать наш сервис по управлению данными большой компании, которая оказывала финансовые услуги», – поясняет Умасанкар Муккара, сооснователь MayaData и недавно созданной Chaos Native. – «Наши клиенты были уверены, что переход в облако пойдет им на пользу – например, ускорит предоставление услуг компании. Но они хотели, чтобы мы доказали надежность продукта.

Муккара и его команда пришли к мнению, что они могли бы использовать методику хаос-инжиниринга для тестирования собственной системы отказоустойчивости, и начали изучать возможные варианты. Они не хотели прибегать к проприетарному решению, а Chaos Monkey не соответствовала их потребностям. «Мы хотели видеть такой продукт, который был бы разработан для работы с Kubernetes», – поясняет Муккара. – «Почти все новое ПО будет написано в облачной среде. Нам нужны такие инструменты, которые бы вписались в эту новую среду».

Таким образом, команда Муккары создала собственную хаос-инжиниринговую платформу, а ее исходный код разместила под названием Litmus. «Открытый исходный код – обычное явление в сегодняшнем мире», – поясняет Муккара. – «Экосистема Kubernetes базируется на свободном, открытом сотрудничестве, поэтому для нас было естественным поделиться нашими наработками с крупнейшими DevOps-сообществами». Об отклике пользователей на этот шаг Муккара говорит: «Реакция была суперпозитивной».

На данный момент проект спонсируют более 100 человек: «Мы получаем обратную связь как от компаний, так и от людей, которые делятся с нами собственным видением того, как сделать сервис более эффективным и улучшить управление жизненным циклом системы».

В свою очередь Эгуаш и его команда подошли к вопросу разработки хаос-инжиниринговой платформы немного по-другому. Они не стали создавать такой инструмент, который был бы приспособлен исключительно к их собственной среде. Вместо этого команда Эгуаша разработала ядро кода, который мог быть использован компаниями из разных сфер, и предоставила им возможность доработать его под свои запросы.

«Мы начали пользоваться Chaos Toolkit потому, что его можно кастомизировать», – говорит Рои Куперман, full-stack разработчик в хостинговой компании Wix. – «Это было легко – создавать отвечающие нашим потребностям плагины или расширять основной проект». Например, Wix внесла функции динамической конфигурации в основной проект Chaos Toolkit.

Именно открытое взаимодействие и возможность каждого внести свой вклад делает открытый исходный код таким важным для хаос-инжиниринга. «Любой человек может написать расширения, которые будут соответствовать конкретно их требованиям», – говорит Эгуаш. – «На

самом деле невозможно, чтобы отдельно взятая компания или открытый исходный код проекта удовлетворили потребности всех и каждого. Я никогда не пользовался Azure, следовательно, не могу создать такой инструмент, который удовлетворял бы потребностям среды Azure. Но сообщество в целом такую способность имеет и может разработать инструменты для любой ИТ-среды».

Хаос повсюду

В сообществе существует четкое понимание того, что хаос-инжиниринг – это всего лишь часть чего-то более масштабного, и что инструменты хаос-инжиниринга не обязательно будут изолированными. Исторически хаос-инжиниринг близок к процессу эксплуатации. Поэтому было бы уместно скоординировать работу по хаос-инжинирингу с функциями этой области, например, с функцией мониторинга.

Стартап Jeli специализируется на анализе инцидентов – близкой к хаос-инжинирингу области, в которой изучаются и делаются выводы из сбоев ИТ систем. «Я думаю, что анализ инцидентов необходим для обеспечения успешной практики хаос-инжиниринга», – говорит Джонс. – «Вам нужно извлечь новую полезную информацию из хаос-экспериментов, и вам нужно создавать новые эксперименты на основе того, что вы поняли, исходя из анализа инцидентов. Сам анализ инцидентов может быть способом понять, какие аспекты работы системы наиболее важны для ваших клиентов, и что вам нужно тестировать еще больше. В моем представлении хаос-инжиниринг и анализ инцидентов идут рука об руку».

Между тем, как и в случае с безопасностью, хаос-инжиниринг смещается влево, к первой половине термина DevOps. «Главное, что мы видим – это внедрение культуры хаоса во всю компанию», – говорит Эгуаш. – «Я ожидаю, что мы увидим такие инструменты хаос-инжиниринга, которые будут поддерживать множество ролей на всем релизном цикле, включая мониторинг и анализ инцидентов, разработку и QA. Сдвиги уже происходят».

В компании Wix хаос-инжиниринг исторически был частью процесса эксплуатации и тесно связан с инструментами мониторинга и оповещения, но он также распространяется и на прикладной уровень. «Я думаю, что хаос-инжиниринг всегда будет основополагающей частью процесса эксплуатации, но в ближайшем будущем он также станет частью процесса CI/CD у тех команд разработчиков, которые заняты созданием инфраструктуры», – говорит Куперман. Компания Harness, которая специализируется на CI/CD, недавно приобрела платформу Chaos Native. Данная сделка подтверждает, что хаос-инжиниринг развивается в направлении сближения с CI/CD.

Открытый исходный код продолжит расширять экспансию хаос-инжиниринга и, что важнее, хаос-мышления в организациях. Каждый из этих проектов предлагает не просто инструменты для проведения хаос-экспериментов, но и определенную инструкцию к тому, как использовать эту дисциплину. «ИТ-индустрии нужно учиться решать серьезные проблемы коллективно. Такими важными проблемами являются безопасность и надежность, с ними сталкиваются все», – говорит Эгуаш. – «Открытый исходный код предоставляет ИТ-сообществу уникальную возможность совместного решения этих проблем».

Вопросы для обсуждения:

- Являются ли инструменты для проведения хаос-испытаний проприетарными или открытыми?
- Как измерить результаты от проведения хаос-испытаний?
- Для каких систем хаос-испытания могут принести наибольшие результаты?
- Что, на Ваш взгляд, ограничивает применение хаос-испытаний в организациях и отдельных командах?
- Найдите в тексте формулировку гипотезы хаос-испытаний. Как ее можно улучшить?
- Как можно проверить такую гипотезу? Что будет результатом проверки?

By Дмитрий Якубовский и Евгений Погребняк