Введение

Вчера начальник попросил вас продемонстрировать клиентам новые замечательные средства, внедряемые в приложение, но вы не смогли показать им ничего. Все ваши разработчики находятся на полпути к финальному релизу, и ни один из них не готов запустить приложение прямо сейчас. У вас есть код, он компилируется и проходит все тесты на вашем сервере непрерывной интеграции (Continuous Integration — CI), но нужно еще несколько дней, чтобы развернуть новую версию в среде приемочного тестирования (User Acceptance Testing — UAT). Имеет ли смысл показывать демонстрационную версию по первому требованию?

В программе обнаружена критическая ошибка. Компания ежедневно теряет на этом деньги. Вы знаете, что нужно исправить: единственную строчку в библиотеке, используемой во всех трех слоях трехуровневой системы. Кроме того, нужно внести соответствующие изменения в таблицу базы данных. Но при поставке предыдущей версии программы эта работа заняла все выходные вплоть до трех часов ночи в понедельник, причем разработчик, осуществлявший развертывание, вскоре уволился, заявив, что этот сумасшедший дом не для него. Вы понимаете, что и теперь за выходные не успеть, а значит, приложение окажется недоступным в течение какого-то периода в рабочие дни. Вряд ли это понравится клиентам компании.

Описанные выше проблемы, хоть и встречаются весьма часто, не являются необходимым следствием цикла разработки программного обеспечения. Скорее, наоборот: они служат признаком того, что в рабочем процессе что-то не так. Поставка очередной версии программы должна быть быстрым, часто повторяющимся процессом. В наши дни многие компании выпускают по несколько версий *ежедневно*! Это вполне возможно даже в очень сложных проектах, содержащих огромный объем кода. В данной книге мы покажем вам, как это делается.

Мэри и Том Поппендик [24] задавались вопросами: “Сколько времени необходимо организации на развертывание новой версии приложения при изменении одной строки кода? Является ли этот процесс повторяющимся и надежным часто?” Интервал времени между принятием решения об изменении кода и поставкой новой версии продукта называется *продолжительностью цикла*. Это важный показатель в любом проекте.

Во многих организациях продолжительность цикла колеблется от нескольких недель до месяцев, причем процесс подготовки релиза не является ни повторяющимся, ни надежным. Часто подготовка выполняется вручную и требует участия команды разработчиков для развертывания программы даже в тестовой и отладочной средах, не говоря уже о рабочей среде. Тем не менее нам часто встречались проекты, которые начинались, как описано выше, однако со временем были существенно доработаны, в результате чего продолжительность цикла была сокращена до нескольких часов и даже минут. Это возможно благодаря созданию полностью автоматизированного, повторяющегося и надежного процесса внесения изменений на различных стадиях сборки, тестирования и развертывания приложений. Ключевой элемент данного процесса — автоматизация, позволяющая буквально одним щелчком на кнопке выполнять разнообразные задачи, связанные с развертыванием программного обеспечения (ПО).

В данной книге описывается, без преувеличения, революционная технология развертывания, сокращающая продолжительность цикла и делающая процесс развертывания надежным и безопасным.

Программный продукт не приносит дохода, пока он не достиг конечного пользователя. Это вполне очевидно. Тем не менее во многих организациях поставка релизов выполняется вручную, в результате чего она чревата ошибками и рисками. Продолжительность цикла все еще измеряется месяцами, а в некоторых организациях она составляет даже больше года, хотя для крупной компании каждая неделя задержки релиза приводит к потере миллионов долларов.

Несмотря на все вышесказанное, механизмы и процессы, позволяющие быстро и без лишних рисков развертывать приложения, все еще не стали обязательной частью проектов, связанных с разработкой ПО.

Наша цель — переход от ручного процесса поставки программного обеспечения к надежному, предсказуемому, контролируемому и максимально автоматизированному процессу с четко понятными и количественно измеряемыми рисками. Применение подхода, описываемого в данной книге, обеспечивает быструю, в течение нескольких часов или даже минут, реализацию идеи в готовом программном продукте высокого качества.

Бо́льшая часть затрат на создание успешного ПО приходится на время после выпуска первой версии. Сюда входит стоимость сопровождения, добавления новых средств и устранения ошибок. Это особенно справедливо для программного обеспечения, поставляемого по итерационной схеме, когда первая версия содержит минимум функциональности, полезной для клиентов. Отсюда название данной книги — *Непрерывное развертывание*, взятое из первого принципа, сформулированного в манифесте гибкой разработки [bibNp0]: “Высшим приоритетом должно быть удовлетворение потребностей заказчика за счет оперативного и непрерывного развертывания надежного программного продукта”. Здесь отражены реалии рынка программного обеспечения: первая версия — это только начало продолжительного процесса поставки успешного приложения.

Все методы, описанные в данной книге, уменьшают время и риски, связанные с развертыванием новых версий программ у конечных пользователей. Эта цель достигается путем налаживания связей и оптимизации взаимодействия команд, занимающихся разработкой, тестированием и поставкой продукта клиентам. Рассматриваемые методы обеспечивают уменьшение интервала времени между модификацией приложения (связанной с добавлением нового средства или исправлением ошибки) и развертыванием очередного релиза на компьютерах пользователей. Кроме того, проблемы можно выявлять на более ранних стадиях и быстрее устранять, а риски, связанные с обновлениями продукта, можно просчитать и минимизировать.

Для кого предназначена книга

Одна из главных целей данной книги — улучшение взаимодействия людей, ответственных за развертывание программного обеспечения, особенно разработчиков, тестировщиков, администраторов баз данных, системных администраторов и менеджеров проектов.

В книге рассматривается широкий круг вопросов, включая управление конфигурацией ПО, контроль исходного кода, планирование выпусков, тестирование, совместимость кодов и автоматизация процессов интеграции, сборки, тестирования и развертывания. Описываются также методы автоматизации приемочного тестирования, управления зависимостями, переноса баз данных, а также создания тестовых и рабочих сред и управления ними.

Многие специалисты, вовлеченные в создание программного обеспечения, считают эти процессы второстепенными по сравнению с написанием кода. Тем не менее наш опыт свидетельствует о том, что они занимают много времени, требуют значительных

усилий и являются критически важными для успешного развертывания ПО. Риски, связанные с этими работами, тяжело поддаются адекватному управлению, в результате чего затраты на них составляют значительную часть стоимости проекта и часто даже превосходят затраты на написание кода. В книге приведена информация, необходимая для понимания этих рисков, и, что еще важнее, описана стратегия их уменьшения.

Это весьма амбициозная цель, и, конечно, все указанные вопросы не могут быть подробно освещены в одной книге. В результате мы рискуем не удовлетворить в полной мере всю нашу потенциальную целевую аудиторию: разработчиков (поскольку в книге не освещены такие важные вопросы, как архитектура ПО, разработка на основе функционирования, и рефакторинг), тестировщиков (в книге недостаточно внимания уделено исследовательскому тестированию и управлению стратегиями тестирования) и администраторов (поскольку недостаточно внимания уделено управлению производительностью, переносу баз данных и мониторингу рабочих процессов).

Мы сознательно пошли на это, потому что существует много книг, в которых подробно рассматриваются все эти темы. Мы считаем, что в существующих книгах не хватает обсуждения того, как совмещаются друг с другом различные этапы технологического процесса, такие как управление конфигурацией ПО, автоматизация тестирования, непрерывные интеграция и развертывание, а также управление данными, рабочими средами и подготовкой релизов. Очень важно оптимизировать все перечисленное в целом, как того требует концепция бережливой разработки. Для этого необходим целостный подход, объединяющий все этапы технологического процесса и всех людей, вовлеченных в него. Начать оптимизацию качества и скорости развертывания ПО можно, только получив полный контроль над продвижением каждого изменения от идеи до выпуска.

Наша цель — выработать целостный подход к проблеме и описать принципы, на которых он основан. Мы предоставим информацию, необходимую для применения принципов непрерывного развертывания в конкретных проектах. Мы не считаем, что существует единственный подход, одинаково пригодный для всех аспектов разработки ПО (не говоря уже о таких сложных вопросах, как управление конфигурацией ПО и информационными процессами в коммерческой среде). Тем не менее описанные в книге фундаментальные принципы непрерывного развертывания применимы в проектах разных типов — больших и малых, долговременных и краткосрочных и т.д.

Начав воплощать эти принципы на практике, вы обнаружите области, для которых необходима более детальная информация. В конце книги приведен список библиографических источников, в которых можно найти подробное объяснение каждой темы, рассматриваемой в книге.

Книга состоит из трех частей. В части I представлены принципы, лежащие “за кулисами” непрерывного развертывания, и методы их практической реализации. В части II рассматривается центральная парадигма книги — шаблон конвейера развертывания. В части III подробно рассматриваются процессы, связанные с поддержкой конвейера развертывания: методы инкрементной разработки, шаблоны управления версиями, управление инфраструктурой, средой и данными, а также общие вопросы управления процессом развертывания.

Многие из описываемых методов применимы только в крупномасштабных проектах. Мы признаем, что имеем опыт работы главным образом с крупными проектами, однако мы убеждены, что рассматриваемые в книге технологии будут полезны и в небольших проектах, особенно если они со временем разрастаются (а это тенденция, присущая большинству проектов). Решения, которые вы принимаете в начале работы над небольшим проектом, существенно влияют на его дальнейшую эволюцию. Правильно начав, вы

убережете себя (и других людей, которые после вас продолжат работу над проектом) от многих осложнений и неприятностей.

Авторы книги исповедуют философию бережливой и итеративной разработки ПО. Под этим мы подразумеваем быстрое и регулярное развертывание надежного ПО, а также непрерывную работу над удалением лишних, ненужных элементов и стадий процесса развертывания. Многие принципы и методы, описанные в книге, первоначально создавались в контексте больших гибких проектов. Тем не менее представленные методы применимы в любых проектах. Внимание в книге сосредоточено на улучшении взаимодействия участников проекта за счет обеспечения открытости и доступности всех частей проекта для всех его участников. Это положительно повлияет на любой проект независимо от того, применяются ли в нем повторяющиеся унифицированные технологии развертывания.

Мы попытались организовать материал книги таким образом, чтобы каждую главу или даже раздел можно было читать независимо от других глав и разделов. Как минимум, мы надеемся, что ссылки на необходимую информацию помогут вам легко найти ее, в результате чего данную книгу можно использовать как справочник.

Важно отметить, что мы не стремились к академической строгости изложения. На рынке есть много серьезных теоретических работ по данной теме, многие из которых могут иметь практический интерес. В частности, мы почти не уделили внимания стандартам, сосредоточившись вместо этого на приемах и методах, полезных для каждого, кто работает над программными проектами. Мы пытались ясно и просто объяснить то, что может быть полезным для повседневной работы. Где это уместно, мы приводим реальные истории из нашей практики для иллюстрации рассматриваемых методов развертывания ПО.

Структура книги

Мы сознаем, что далеко не каждый человек прочитает эту книгу от корки до корки. Книга написана таким образом, что приступать к ее чтению можно по-разному. Некоторые положения повторяются в разных местах книги; естественно, не до такой степени, чтобы книга стала скучной, если вы решите прочитать ее от начала до конца.

Книга состоит из трех частей. В части I (главы 1–4) рассматриваются базовые принципы подготовки повторяющихся релизов с минимальными рисками и методы их поддержки. В части II (главы 5–10) описывается конвейер развертывания. Начиная с главы 11 рассматривается рабочая среда непрерывного развертывания.

Мы рекомендуем прочитать главу 1 каждому читателю. Нам кажется, что в ней есть много полезной информации как для новичков в развертывании ПО, так и для опытных разработчиков. Вы найдете в ней идеи, которые изменят ваш взгляд на профессиональную разработку ПО. Остальные главы книги можно читать либо для повышения профессионального уровня в свободное от работы время, либо при возникновении острой необходимости решить конкретную проблему.

Часть I. Основы непрерывного развертывания

В части I приведена информация, необходимая для понимания концепции конвейера развертывания. Каждая следующая глава тематически связана с предыдущей.

Глава 1, “Проблема развертывания программного обеспечения”, начинается с рассмотрения ряда “стихийных”, неэффективных шаблонов развертывания, которые можно встретить во многих проектах. На их примере мы объясняем цели данной книги и методы

их достижения. Глава завершается формулировкой принципов развертывания, которым посвящены остальные главы книги.

В главе 2, “Стратегии управления конфигурациями”, рассматривается взаимодействие процессов сборки, тестирования и развертывания приложения, начиная с исходного кода и сценариев сборки и заканчивая конфигурациями рабочей среды и приложения.

Глава 3, “Непрерывная интеграция”, посвящена построению и запуску автоматизированных тестов после каждого изменения исходного кода приложения. Особое внимание уделяется постоянному поддержанию приложения в рабочем состоянии.

В главе 4, “Реализация стратегии тестирования”, представлены различные процедуры ручного и автоматизированного тестирования, составляющие неотъемлемую часть каждого проекта. Кроме того, в главе обсуждается принятие решения о стратегии проекта, обеспечивающей надежное развертывание.

Часть II. Конвейер развертывания

В части II подробно рассматривается конвейер развертывания, включая реализацию отдельных стадий конвейера.

В главе 5, “Структура конвейера развертывания”, представлен главный шаблон книги — автоматизированный процесс продвижения изменений от идеи до выпуска. В частности, обсуждается реализация конвейера на организационном и командном уровнях.

Глава 6, “Сценарии сборки и развертывания”, посвящена технологиям создания сценариев развертывания, которые можно использовать для автоматической сборки приложений. Приводятся рекомендации по их наиболее эффективному использованию.

В главе 7, “Стадия фиксации”, рассматривается первая стадия конвейера — набор автоматизированных процессов, запускаемых в момент внесения любого изменения в программный продукт. Кроме того, обсуждается создание быстрых и эффективных наборов тестирования.

Глава 8, “Автоматическое приемочное тестирование”, посвящена анализу и практической реализации систем приемочного тестирования. Обсуждаются тесты, необходимые для непрерывного развертывания, и рассматривается создание наборов тестирования, защищающих функциональность приложения на разных стадиях конвейера.

В главе 9, “Тестирование нефункциональных требований”, главное внимание уделено тестированию производительности, хотя вкратце обсуждаются и другие нефункциональные требования. Подробно рассматривается создание тестов производительности и конфигурирование среды тестирования.

В главе 10, “Развертывание и выпуск приложений”, мы рассмотрим, что происходит после стадии автоматизированного тестирования, а именно: как происходит перенос релиз-кандидатов в среду ручного тестирования, затем в среду приемочного тестирования и в отладочную среду вплоть до окончательной поставки релиза. Будет показано, как осуществляется непрерывное развертывание, как происходят откаты и как создаются релизы с нулевым временем простоя.

Часть III. Процесс поставки

В заключительной части книги обсуждаются комплексные методы и технологии поддержки конвейера развертывания.

Глава 11, “Управление инфраструктурой и средами”, посвящена автоматизации процессов создания, конфигурирования и мониторинга тестовых сред, включая применение облачных вычислений и виртуализации.

В главе 12, “Управление данными”, рассматривается создание и перенос тестовых и рабочих данных между разными стадиями жизненного цикла приложения.

Глава 13, “Управление компонентами и зависимостями”, начинается с рассмотрения методов непрерывного поддержания приложения в состоянии, пригодном для быстрой подготовки повторяющихся релизов. Затем описываются принципы организации приложения в виде коллекции элементов и управления их сборкой и тестированием.

В главе 14, “Управление версиями”, приведен обзор наиболее популярных инструментов управления версиями. В этой же главе подробно описываются разные шаблоны управления версиями.

В главе 15, “Управление непрерывным развертыванием”, описаны подходы к управлению рисками и совместимостью, а также представлена модель зрелости процессов управления конфигурациями и поставкой. Кроме того, мы поговорим о ценности непрерывного развертывания для коммерческих приложений и рассмотрим жизненный цикл итеративных проектов развертывания.

Веб-ссылки в книге

Как правило, в книге приводятся не полные, а сокращенные адреса внешних сайтов, например `[bibNp0]`. Открыть такую ссылку одним из двух способов. Первый из них предполагает использование службы преобразования адресов, доступной на сайте `bit.ly`. В таком случае адрес для рассматриваемого ключа будет выглядеть так: `http://bit.ly/bibNp0`. Второй способ основан на использовании службы, которую мы установили по адресу `http://www.continuousdelivery.com/go/`. Она использует те же самые ключи, поэтому полный адрес в данном случае будет выглядеть так: `http://www.continuousdelivery.com/go/bibNp0`. Мы поддерживаем службу на тот случай, если сайт `bit.ly` по какой-либо причине окажется недоступным. Если изменится адрес веб-страницы, на которую дана ссылка, мы постараемся обновить базу данных на сайте `http://www.continuousdelivery.com/go`, так что обращайтесь к нему, если вдруг ссылки на сайте `bit.ly` не работают.

Изображение на обложке

Все книги серии Мартина Фаулера “Signature Series” содержат изображение моста на обложке. Первоначально мы собирались использовать фотографию Железного Моста в Англии, но она уже оказалась выбрана для другой книги серии. Тогда мы решили выбрать другую британскую достопримечательность: железнодорожный мост через Фертоф-Форт, представленный на великолепной фотографии Джорджа Гастина.

Это был первый стальной мост в Великобритании. Сталь отливали на двух сталелитейных заводах в Шотландии и одном — в Уэльсе с использованием новых для того времени мартеновских печей. Сталь доставлялась в виде готовых трубчатых ферм — впервые в Великобритании при строительстве моста использовались серийные детали. Проектировщики моста, сэр Джон Фаулер, сэр Бенджамин Бейкер и Аллан Стюарт, просчитали влияние монтажных напряжений, предусмотрели сокращение будущих эксплуатационных расходов, а также провели расчеты ветровой нагрузки и температурных напряжений в конструкции. Все это напоминает нам анализ функциональных и нефункциональных требований к программному обеспечению. Проектировщики лично контролировали возведение моста, дабы убедиться в соблюдении всех требований.

На строительстве моста было задействовано свыше 4600 рабочих, из которых более полусотни погибло и несколько сот получили увечья. Тем не менее конечный результат представляет собой один из шедевров промышленной революции: на момент завершения строительства в 1890 году это был самый длинный мост в мире, и даже в начале 21 века он является вторым по длине консольным мостом в мире. Подобно долгоиграющему программному проекту, мост требует постоянного обслуживания. Это было изначально продумано в проекте, что вылилось в строительство не только ремонтной мастерской и сортировочной станции, но и целого железнодорожного депо из примерно пятидесяти зданий неподалеку от железнодорожной станции Далмени. Оставшийся срок эксплуатации моста оценивается в 100 с лишним лет.