Господь благословил меня замечательной семьей.
Посвящается моим родителям, Полю и Ноне, моим братьям и сестрам, Сью, Джоан, Джону, Мэри, Салли, Тиму, Полин и Эви.
П.М.Д.

Введение

Предисловие Мартина Фаулера[1]

В прежние времена одним из наиболее сложных и напряженных моментов проектирования программного обеспечения была интеграция. При сборке воедино модулей, разработанных по отдельности, зачастую возникают проблемы, обнаружить причины которых бывает крайне сложно. Однако за последние несколько лет интеграция почти перестала быть головной болью проекта, во всяком случае теперь это уже *не событие*.

Причиной такого превращения является приобретение навыков интеграции. С одной стороны, ежедневное построение можно считать амбициозной целью, с другой — в большинстве известных мне проектов интеграция осуществлялась по несколько раз в день. Как ни парадоксально, но когда встречается некое затруднительное действие, хороший совет — делать его почаще.

Самым интересным в непрерывной интеграции является то, что людей зачастую удивляет ее влияние. Нередко люди отказываются от нее как от маргинальной возможности, хотя это может полностью изменить стиль проекта. Однако смысл в этом намного больший, ведь так проблемы обнаруживаются быстрее. Поскольку времени между возникновением и обнаружением ошибки проходит меньше, ее проще найти, тем более что можно легко просмотреть сделанные изменения и отыскать источник проблем. Вместе с соответствующей программой проверки это может привести к существенному уменьшению количества ошибок. В результате разработчики тратят меньше времени на отладку, больше занимаются реализацией возможностей и сохраняют уверенность в надежности создаваемого.

Безусловно, простого совета интегрировать чаще недостаточно, поскольку за этим кроется набор принципов и практический опыт, позволяющие сделать непрерывную интеграцию действительностью. Многое из упомянутого здесь можно найти рассеянным по другим книгам и Интернету (и я горд, что принял участие в создании этого содержимого), но все это придется искать самостоятельно.

Поэтому я очень рад, что Поль собрал всю эту информацию в одну книгу, настоящий справочник для желающих получить полезный совет. Подобно любой практике наибольшие проблемы кроются в деталях. За последние несколько лет мы многое узнали об этих составляющих и научились справляться с проблемами. Эта книга обобщает накопленный опыт, предоставляя надежную основу знаний о непрерывной интеграции, а также раскрывает ее значение для разработки программного обеспечения.

Предисловие Пола Джулиуса

Я был уверен, что рано или поздно кто-нибудь напишет подобную книгу. По секрету, я всегда надеялся, что это буду я. Но я доволен, что Поль, Стив и Энди наконец связали все это воедино в продуманный трактат.

Я всегда занимался чем-то похожим на непрерывную интеграцию. В марте 2001 года я участвовал в основании проекта с открытым исходным кодом CruiseControl, будучи его администратором. В моей повседневной работе я постоянно консультировался с консал-

[1] Мартин Фаулер — редактор серии и главный научный сотрудник компании ThoughtWorks.

тинговой компанией ThoughtWorks, помогающей в создании клиентских структур, построении и развертывании решений проверки с использованием принципов и инструментов непрерывной интеграции.

Работая с почтовыми сообщениями проекта CruiseControl до 2003 года, я получил возможность прочитать описания тысяч различных случаев непрерывной интеграции. Проблемы, с которыми сталкиваются создатели программного обеспечения, весьма разнообразны и сложны. Со временем я все яснее видел причины, по которым разработчики мирились с этими сложностями. Преимущества непрерывной интеграции, такие как быстрая обратная связь, быстрое развертывание и воспроизводимая автоматизированная проверка, существенно их превышали. Однако как просто зачастую пропустить метку при создании типов окружения. Я никогда не предполагал, когда мы впервые выпустили CruiseControl, насколько захватывающи способы, которыми будет использоватся непрерывная интеграция для улучшения процесса разработки программного обеспечения.

В 2000 году я работал над большим проектом приложения J2EE, используя все возможности, предоставляемые данной спецификацией. Приложение было в своем роде уникальным, но создавалось сложно. Под построением я подразумеваю компиляцию, проверку, архивирование и проверку функций. Система Ant все еще пребывала в младенчестве и еще не стала фактическим стандартом для приложений Java. Для компиляции всего необходимого и запуска проверки модулей мы просто использовали ряд сценариев оболочки. Для передачи всего в разворачиваемый архив применялся еще один набор сценариев оболочки. И наконец, мы вручную выполняли ряд операций по развертыванию файлов JAR и запуску комплекта проверки функций. Само собой разумеется, этот процесс был трудоемок, утомителен и чреват ошибками.

Так начались мои поиски перенастраиваемого средства "построения", для которого было бы достаточно нажать "одну кнопку" (одна из любимых тем Мартина Фаулера). Система Ant решила проблему создания независимых от платформы сценариев построения. Оставалось найти что-нибудь для выполнения таких утомительных действий, как развертывание, проверка функций и сообщение о результатах. Я исследовал существующие решения, но напрасно. Работая над тем проектом, я так и не нашел достаточно работоспособного средства, которое подходило бы мне полностью. Разработка приложения завершилась успешно, но я знал, что все можно было сделать проще и лучше.

Ответ я отыскал между завершением того проекта и началом следующего. Мартин Фаулер и Матт Фоеммел (Matt Foemmel) как раз опубликовали свою фундаментальную статью о непрерывной интеграции. Случайно я познакомился с людьми из ThoughtWorkers, работавшими над реализацией системы многократного использования решения Фаулера и Фоеммела. Я был несказанно рад! Я знал, что это ответ на все мои вопросы, оставшиеся от предыдущего проекта. Через несколько недель у нас уже все было готово, и мы применили данный подход в нескольких существующих проектах. Я даже посетил сайт бета-тестирования предшественника системы CruiseControl, чтобы опробовать ее в полном объеме. Вскоре после этого мы перешли на реализацию с открытым исходным кодом. Для меня уже не было никакого смысла оглядываться назад.

В качестве консультанта в ThoughtWorks я сталкивался с некоторыми из наиболее сложных случаев развертывания корпоративных архитектур. Нашим клиентам зачастую требовалось быстрое решение на основании преимуществ высокоуровневых концепций, обещанных в литературе. Подобно любой технологии, в них есть приличная доза дезинформации о том, насколько легко все это внедрить на предприятии. Если годы консультаций и научили меня чему, так это тому, что ничто не является столь же простым, как кажется.

Мне нравится разговаривать с клиентами о реальном применении принципов непрерывной интеграции. Я предпочитаю подчеркнуть важность перевода разработки “интонацией”, чтобы корректно обозначить преимущества. Когда разработчики проводят проверки только раз в месяц, не уделяя внимания автоматизированному контролю, или не имеют никакой заинтересованности в немедленном устранении ошибок, их ожидают большие неприятности, для решения которых понадобятся все преимущества непрерывной интеграции.

Означает ли это, что менеджеры IT не должны ничего знать о непрерывной интеграции до тех пор, пока дело не дойдет до проблем? Вовсе нет. Фактически практическое применение непрерывной интеграции может быть одним из наиболее существенных мотивов для изменений. Я полагаю, что установка таких инструментов непрерывной интеграции, как CruiseControl, мотивирует активность группы разработчиков программного обеспечения. Изменение не происходит внезапно, и вы должны соразмерить ваши ожидания соответственно (это относится и к менеджерам IT). При настойчивости и хорошем понимании основных принципов даже наиболее сложные системы могут быть созданы простыми и понятными, легкими для проверки и быстрой разработки.

Этой книгой авторы подготовили хорошее игровое поле. Я полагаю, что она будет не только исчерпывающей, но и весьма популярной долгое время. Глубокий анализ наиболее важных аспектов непрерывной интеграции поможет читателям принимать хорошо продуманные решения. Диапазон рассматриваемых тем обширен и включает описание доминирующих ныне подходов непрерывной интеграции, помогая читателям взвесить компромиссы, на которые они должны будут пойти. И наконец, мне нравится наблюдать, как задачи, к достижению которых стремятся столь многие в сообществе непрерывной интеграции, постепенно формализуются и становятся основой для дальнейших исследований. Поэтому я настоятельно рекомендую эту книгу как жизненно важный ресурс для разработчиков корпоративных приложений в условиях сложной географии, использующих магию непрерывной интеграции.

Предисловие

В начале моей карьеры в одном из журналов я увидел анонс на всю страницу, где был изображен фрагмент клавиатуры с клавишей, подобной клавише <Enter> и помеченной словом “Integrate” (Интегрировать) (рис. П.1). Текст ниже гласил: “Если бы только это было так просто”. Не помню, кто и для чего привел эту иллюстрацию, но мысль мне понравилась. Однако в свете разработки программного обеспечения я полагал, что такое никогда не будет возможным, поскольку в нашем проекте мы провели несколько дней в “аду интеграции”, пытаясь собрать воедино бесчисленные компоненты программного обеспечения в конце каждого промежуточного этапа проекта. Однако концепция мне понравилась настолько, что я вырезал рисунок и повесил его у себя на стенке. Одной из главных задач эффективной разработки программного обеспечения мне представлялась автоматизация повторяемых процессов и процессов, склонных к ошибкам. Кроме того, это воплощало мое убеждение в возможности сделать интеграцию программного обеспечения “не событием” проекта (как это назвал Мартин Фаулер), а неким обычным действием. *Непрерывная интеграция* (Continuous Integration — CI) способна превратить это в реальность.

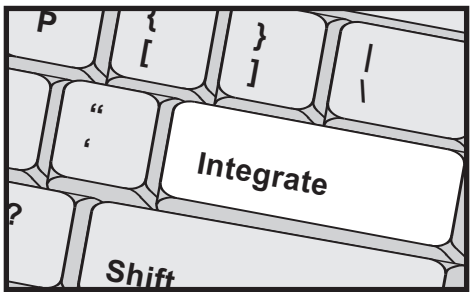

Рис. П.1 Кнопка <Integrate>!

О чем эта книга

Здесь описаны некоторые из наиболее типичных процессов разработки программного обеспечения: компиляция кода, определение данных и манипулирование ими в базе данных; осуществление проверки, просмотр кода и, в конечном счете, развертывание программного обеспечения. Кроме того, участники группы, безусловно, должны общаться друг с другом при обсуждении состояния программного обеспечения. Только представьте, что все эти процессы можно запустить, нажав всего лишь одну кнопку.

В книге рассматривается создание виртуальной кнопки <Integrate>, автоматизирующей большинство процессов разработки программного обеспечения. Но главное, в ней описано, как такая кнопка способна снизить риски, подстерегающие при построении приложений, например, от позднего обнаружения дефектов и низкокачественного кода. В системе CI большинство этих процессов автоматизировано, и они запускаются после каждого изменения разрабатываемого программного обеспечения.

Что такое непрерывная интеграция?

Процесс интеграции программного обеспечения — далеко не новая проблема. В проекте, выполняемом одним человеком с немногими внешними зависимостями, интеграция программного обеспечения — не слишком существенная проблема, но при увеличении сложности проекта (даже если в него просто добавлен еще один человек) возникает насущная потребность в интеграции и проверке слаженной работы компонентов программного обеспечения, причем заранее и *часто*. Дожидаться конца проекта для проведения интеграции и выявления всего спектра возможных ошибок — неразумно и к тому же не способствует качеству программного обеспечения, а зачастую даже приводит к удорожанию и задержке сдачи проекта. Непрерывная интеграция снижает подобные риски.

В своей популярной статье “Continuous Integration”[2] Мартин Фаулер описывает CI так:

... практика разработки программного обеспечения, когда участники группы осуществляют частую интеграцию своих работ. Обычно каждый человек проводит интеграцию по крайней мере ежедневно, что приводит к нескольким интеграциям в день. Для максимально быстрого обнаружения ошибок каждая интеграция осуществляется автоматизированно (вместе с проверкой). Многие группы находят, что данный подход позволяет значительно уменьшить проблемы интеграции и способствует более быстрой разработке программного обеспечения.

[2] См. `www.martinfowler.com/articles/continuousIntegration.html`.

А это означает следующее:

- все разработчики выполняют закрытое построение[3] на собственных рабочих станциях перед передачей кода в хранилище с контролем версий, для гарантии того, что внесенные изменения не приведут к ошибке при интеграционном построении;
- разработчики обновляют свой код в хранилище с контролем версий *по крайней мере* один раз в день;
- интеграционное построение осуществляется несколько раз в день на выделенной для этого машине;
- при каждом построении проводится 100 % проверок;
- создаваемый продукт (например, файл WAR, сборка, исполняемый файл и т.д.) пригоден для функциональной проверки;
- исправление ошибок имеет самый высокий приоритет;
- часть разработчиков просматривают отчеты, созданные в ходе построения, стандарты программирования и отчеты анализа зависимостей, в поисках областей для усовершенствования.

В этой книге обсуждаются аспекты автоматизации CI, большинство предоставляемых ей преимуществ в области повторяемых и склонных к ошибкам процессов; однако Фаулер идентифицирует CI как процесс часто осуществляемый, но не как автоматизированный. Однако, поскольку ныне существует множество великолепных инструментальных средств, поддерживающих CI как автоматизированный процесс, использующий сервер CI для автоматизации действий CI — данный подход эффективен. Тем не менее ручной подход к интеграции (при автоматизированной компиляции) вполне может хорошо сработать в любой группе.

Быстрая обратная связь

Непрерывная интеграция увеличивает возможности обратной связи. Она позволяет следить за состоянием проекта в течение дня. CI применяется для уменьшения временного промежутка между моментом проявления дефекта и его устранением, улучшая таким образом общее качество программного обеспечения.

Группе разработки вовсе не следует полагать, что автоматизация системы CI избавила их от проблем интеграции. Это тем более справедливо, если группа использует автоматизированный инструмент не более чем для компиляции исходного кода; некоторые именно это считают “построением” (build), что совсем не так (см. главу 1, “Первые шаги”). Эффективная практика CI подразумевает намного больше, чем применение соответствующих инструментов. Сюда относятся действия, которые будут описаны в этой книге, например частые обновления файлов в хранилище с контролем версий, немедленное устранение проблем, а также использование отдельной машины для интеграционного построения.

Практика CI обеспечивает более быструю обратную связь. Применение эффективных практик CI позволяет узнавать общее состояние разрабатываемого программного обеспечения *по нескольку раз в день*. Более того, CI хорошо работает с такими практиками, как рефакторинг и разработка методом проверки, поскольку в их основе лежит концепция небольших изменений. В сущности, CI гарантирует совместимость последних с остальной

[3] Схемы закрытого (системного) и интеграционного построения описаны в книге *Software Configuration Management Patterns* Стивена П. Беркзука (Stephen P. Berczuk) и Бреда Апплетона (Brad Appleton).

частью программного обеспечения. На более высоком уровне CI повышает коллективную ответственность группы и снижает трудоемкость проекта, поскольку уменьшает объем *ручного* труда, выполняемого при каждом изменении разрабатываемого программного обеспечения.

Замечание о слове "непрерывное"

В этой книге используется термин "непрерывная" ("continuous"), однако его употребление технически неправильно. *Непрерывность* подразумевает нечто, что, будучи начато один раз, никогда не завершается. Это предполагает непрерывность процесса интеграции, а этого нет даже в наиболее интенсивной системе CI. Таким образом, в этой книге описывается то, что скорее является "очень частой интеграцией".

Для кого написана эта книга

Опыт свидетельствует, что существует отличие между теми, кто рассматривает разработку программного обеспечения как *задачу*, и теми, кто считает ее своей *профессией*. Эта книга для тех, кто работает профессионально и выполняет в проекте повторяемые процессы. В ней описываются практики и преимущества CI, а также предоставляется информация о том, как их применять для высвобождения времени и накопления опыта для решения более важных проблем.

Данная книга раскрывает основные темы, связанные с CI, включая реализацию последней с использованием непрерывной обратной связи, проверки, развертывания, инспекции и интеграции базы данных. Независимо от того, какова роль читателя в разработке программного обеспечения, можно включать элементы CI в собственные процессы построения. Если вы профессионал-разработчик, желающий повысить эффективность своей работы, т.е. делать больше за то же время и с более предсказуемым результатом, то эта книга для вас.

Разработчики

Если вы заметили, что вместо разработки программного обеспечения для пользователей занимаетесь решением проблем интеграции, то эта книга поможет избавиться от них и вернуться к основной работе. Настоящая книга не требует больших временны*х* затрат на интеграцию; она, наоборот, о том, как облегчить интеграцию программного обеспечения и позволить сосредоточиться на любимом занятии — разработке. Многие практики и примеры, приведенные в книге, демонстрируют способы реализации эффективной системы CI.

Управление построением, настройкой и выпуском

Если ваша задача заключается в выпуске *работоспособного* программного обеспечения, то вы найдете эту книгу особенно интересной, поскольку в ней демонстрируется, что запуск процессов при *каждом* внесении изменений в хранилище с контролем версий позволит создавать слаженное работоспособное программное обеспечение. Большинство руководителей проекта выполняют и другие функции в процессе построения, например разработку. Система CI сможет самостоятельно решить некоторые из подобных задач, тем самым избавив от необходимости ждать конца цикла разработки, чтобы получить развертываемое и *проверяемое* (testable) программное обеспечение.

Испытатели

Непрерывная интеграция обеспечивает быструю обратную связь при разработке программного обеспечения, но не устраняет традиционных повторно встречающихся дефектов даже после запуска "фиксации". Испытатели обычно получают удовлетворение от своей роли в проекте, где применяется CI, поскольку программное обеспечение чаще доступно для проверки, к тому же с меньшими проверяемыми областями. При использовании системы CI в цикле разработки контролируется *все и всегда*, в отличие от традиционного сценария, где испытатели либо проверяют все в последние часы или не проверяют вовсе.

Менеджеры

Эта книга может произвести на вас впечатление, если вы ищете высоко компетентное издание для группы, регулярно и неоднократно выпускающей работоспособное программное обеспечение. Вы можете контролировать сроки, бюджет и качество куда эффективнее, поскольку будете основывать свои решения по рабочему программному обеспечению на фактической обратной связи и показателях, а не только на элементах задачи и расписании проекта.

Структура книги

Данная книга разделена на две части. Первая содержит введение в CI, исследующее ее концепции и практики. Эта часть адресована тем читателям, которые не знакомы с базовыми практиками CI. Хотя, безусловно, на этом практики CI не исчерпываются. Вторая часть развертывает базовые концепции в набор эффективных процессов, выполняемых системами CI, такими как проверка, инспекция, развертывание и обратная связь.

Часть I. Основы CI — принципы и практики

Глава 1, "Первые шаги", сразу вводит в курс дела высокоуровневым примером использования сервера CI при непрерывном построении программного обеспечения.

Глава 2, "Введение в непрерывную интеграцию", знакомит с общими практиками и способами применения CI.

Глава 3, "Снижение риска с использованием CI", обсуждает ключевые риски, которые может смягчать CI, на примере использования сценариев.

Глава 4, "Построение программного обеспечения при каждом изменении", исследует практику интегрирования программного обеспечения при каждом изменении с применением автоматизированной компиляции.

Часть II. Создание полнофункциональной системы CI

Глава 5, "Непрерывная интеграция баз данных", переходит к более сложным концепциям, включая процесс перестройки базы данных и применения проверки последних в качестве частей интеграционного построения.

Глава 6, "Непрерывная проверка", рассматривает концепции и стратегии проверки программного обеспечения при каждом интеграционном построении.

Глава 7, "Непрерывная инспекция", знакомит с использованием различных инструментальных средств и методов автоматизированной, а также непрерывной инспекции (статический и динамический анализ).

Глава 8, “Непрерывное развертывание”, исследует процесс развертывания с применением системы CI, обеспечивающей функциональную проверку.

Глава 9, “Непрерывная обратная связь”, описывает и демонстрирует использование устройств непрерывной обратной связи (например, электронной почты, RSS, X10 и Ambient Orb), способных уведомить об успехе или отказе построения.

В эпилоге описаны будущие возможности CI.

Приложения

Приложение А, “Ресурсы CI”, включает список URL, инструментов и документов о CI.

Приложение Б, “Обсуждение инструментальных средств CI”, содержит оценку различных серверов CI и соответствующих инструментальных средств, доступных на рынке. Здесь обсуждается их применимость для практик, описанных в книге, анализируются их преимущества и недостатки, а также объясняются методы использования некоторых из них.

Другие средства

В книге содержатся следующие элементы, призванные облегчить изучение изложенного материала.

- Практики. В настоящей книге рассматривается более сорока практик, относящихся к CI. Множество подзаголовков глав относятся к действиям. Рисунки в начале большинства глав иллюстрируют описываемые в ней практики и позволяют просматривать интересующие области. Например, *используйте выделенную машину для интеграционного построения* и *обновляйте код чаще* — вот два примера практик, обсуждаемых в книге.
- Примеры. На основе различных примеров на разных языках и платформах будет продемонстрировано, как применять эти практики.
- Вопросы. Каждая глава заканчивается списком вопросов, которые помогут оценить пригодность практики CI для конкретного проекта.
- Web-сайт. Web-сайт `www.integratebutton.com` предоставляет обновления, примеры кода и другие материалы для поддержки этой книги.

Что можно узнать

Читая эту книгу, можно изучить концепции и практики, позволяющие создавать слаженное, работоспособное программное обеспечение. Мы прежде всего сосредоточимся на практиках, иллюстрируя способы их применения наглядными примерами везде, где это возможно. В примерах использованы разные платформы разработки, такие как Java, Microsoft .NET и даже Ruby. В качестве основного сервера CI на протяжении всей книги используется CruiseControl (версии Java и .NET); однако мы создавали подобные примеры, используя также и другие серверы и инструменты; они перечислены на сопутствующем Web-сайте (`www.integratebutton.com`) и в приложении Б, “Обсуждение инструментальных средств CI”.

Читая эту книгу, можно научиться следующему:

- как реализовать CI, создающую *развертываемое программное обеспечение* на каждом этапе разработки;

- как CI способна *сократить время* между проявлением дефекта и обнаружением его причины, удешевляя таким образом его устранение;
- как обеспечить *качество программного обеспечения* в результате его частой интеграции, не откладывая это на последние этапы разработки.

Что не рассматривается в данной книге

Эта книга не рассматривает подробно каждый инструмент (планирование построения, среду программирования, контроль версий и т.д.), составляющий систему CI. Ее основное внимание уделено реализации практик CI, призванных выработать эффективную систему CI. Сначала обсуждаются практики CI; если некий рассматриваемый инструмент больше не используется или не соответствует текущим потребностям, просто применяйте практику, используя другой инструмент, чтобы получить тот же результат.

Кроме того, невозможно и бесполезно описывать каждый тип проверки, механизм обратной связи, автоматизированный инспектор или тип развертывания, используемый системой CI. Мы надеемся, что сосредоточившись на диапазоне ключевых практик, предоставив примеры применения методов и инструментов для интеграции баз данных, проверки, инспекции и обратной связи, мы выполнили свою основную задачу и вдохновили вас на создание приложений, столь же великолепных, как и проекты ведущих групп. Как неоднократно упоминается в книге, сопутствующий Web-сайт `www.integratebutton.com` содержит примеры, инструментальные средства и языки, которые не могут быть описаны в настоящей книге.

Авторство

Эта книга — результат совместных усилий трех авторов и одного соавтора. Я написал большинство глав. Стив Матиас занимался главами 4, 5, 7, 8 и приложением A, а также некоторыми из примеров книги. Перу Энди Гловера принадлежат главы 6, 7 и 8, он также предоставил примеры и внес вклад в другие элементы книги. Эрик Тавела написал приложение B. Таким образом, когда в предложениях используются сентенции от первого лица, следует иметь в виду конкретного автора, которому принадлежат эти слова.

Об обложке

Я был невероятно обрадован, узнав, что наша книга войдет в известную серию книг Мартина Фаулера. Поэтому я решил выбрать для обложки книги мост. Мои соавторы и я относимся к редкой породе людей, выросших в районе Вашингтона, округ Колумбия. Тем, кто вырос не в нашем округе, заметим, что эта область богата разнообразными переходами. Конкретней, мы из Северной Вирджинии, поэтому, отдавая дань этому факту, мы выбрали для обложки Природный Мост Вирджинии (Natural Bridge in Virginia). До этого я никогда не посещал его. Он имеет очень интересную историю, и мне показалось невероятным, что это полнофункциональный мост, по которому автомобили проезжают каждый день. (Безусловно, я проехал на машине по нему несколько раз.) Мне хотелось бы надеяться, что прочитав эту книгу, вы сделаете CI естественной частью вашего следующего рабочего проекта программного обеспечения.

Благодарности

Я много раз читал благодарности авторов типа “Мне никогда бы не удалось это сделать самому” и другие вещи в этом роде. Я всегда полагал, что это всего лишь ложная скромность. Но, как оказалось, я был не прав. Эта книга была титаническим свершением, которое я осуществил благодаря людям, перечисленным ниже.

Я хотел бы поблагодарить мое издательство, Addison-Wesley, и в частности выразить благодарность моему исполнительному редактору, Крису Гузикоуски (Chris Guzikowski), за его работу со мной в течение этого изнурительного процесса. Его опыт, понимание и поддержку трудно переоценить. Кроме того, мой научный редактор, Крис Зан (Chris Zahn), обеспечивал компетентные рекомендации на протяжении нескольких версий и циклов редактирования. Я также хотел бы выразить искреннюю признательность Карен Геттман (Karen Gettman), Мишель Хаусли (Michelle Housley), Джессике Д'Амико (Jessica D’Amico), Джули Нэйхил (Julie Nahil), Ребекке Гринберг (Rebecca Greenberg) и наконец, но не в последнюю очередь, моему первому исполнительному редактору Мэри О'Брайен (Mary O’Brien).

Рич Миллс (Rich Mills) предоставил для книги сервер CVS и выдавал превосходные идеи на сеансах мозгового штурма. Я также хотел бы поблагодарить моего наставника и друга Роба Дали (Rob Daly) за то, что он приобщил меня в 2002 году к профессиональному писательству, попросив составить исключительно подробные обзоры процесса программирования. Джон Стивен (John Steven) вдохновил меня на написание этой книги.

Я хотел бы выразить признательность моим соавторам, редакторам и помощникам. Со Стивом Матиасом мы провели множество бессонных ночей, создавая то, что вы читаете сегодня. Энди Гловер был нашим координатором, предоставившим свой обширный опыт разработки и проверки в процессе проектирования.

Лайза Портер, наш редактор-соавтор, неустанно совершенствовала каждый вариант книги, редактируя и предоставляя рекомендации, которые помогли улучшить ее качество. Благодарю Эрика Тавела, написавшего приложение по инструментам CI, и Левента Гарсес (Levent Gurses) за предоставленный опыт по Maven 2 в приложении B.

У нас были и электронные помощники, а также персональные технические рецензенты, поддерживавшие мгновенную обратную связь на протяжении всего проекта. Среди них Том Копланд (Tom Copeland), Роб Дали, Салли Дюваль (Sally Duvall), Каспер Хорнструп (Casper Hornstrup), Джо Хант (Joe Hunt), Айрин Джексон (Erin Jackson), Джо Кониор (Joe Konior), Рич Миллс, Лесли Пауер (Leslie Power), Дэвид Сиск (David Sisk), Карл Таллис (Carl Tallis), Эрик Тавела, Дэн Тейлор (Dan Taylor) и Сажит Васудеван (Sajit Vasudevan).

Я также хотел бы благодарить Чарльза Марри (Charles Murray) и Кристалл Белония (Cristalle Belonia) за помощь, а также Масиеж Завадски (Maciej Zawadzki) и Эрика Миник (Eric Minick) за помощь с Urbancode.

Я благодарю за поддержку многих прекрасных людей, вдохновлявших меня каждый день, а именно: Бурк Кокс (Burke Cox), Мэнди Оуенс (Mandy Owens), Дэвида Вуда (David Wood) и Рона Райта (Ron Wright). Существуют также люди, вдохновлявшие меня на протяжении многих лет, среди которых Рич Кемпбелл (Rich Campbell), Дэвид Фадо (David Fado), Майк Фрейзер (Mike Fraser), Брент Гендлеман (Brent Gendleman), Джон Хьюз (Jon Hughes), Джеф Хванг (Jeff Hwang), Черри Хванг (Sherry Hwang), Сэнди Кайли (Sandi Kyle), Брайан Лионс (Brian Lyons), Сьюзен Мейсон (Susan Mason), Брайан Мессер (Brian Messer), Сэнди Миллер (Sandy Miller), Джон Ньюман (John Newman), Маркус Оуэн (Marcus Owen), Крис Паинтер (Chris Painter), Полетт Роджерс (Paulette Rogers), Марк Симоник (Mark Simonik), Джо Стасник (Joe Stusnick) и Майк Траил (Mike Trail).

Я также ценю помощь группы технических рецензентов издательства Addison-Wesley, включая Скотта В. Эмблера (Scott Ambler), Бреда Апплетона (Brad Appleton), Джона Эвеса (Jon Eaves), Мартина Фаулера, Пола Холсера (Paul Holser), Пола Джулиуса (Paul Julius), Кирка Кноршильда (Kirk Knoernschild), Майка Мелия (Mike Melia), Джулиан Симпсон (Julian Simpson), Энди Тригг (Andy Trigg), База Водд (Bas Vodde), Майкла Вард (Michael Ward) и Джейсон Уип (Jason Yip).

Я хочу поблагодарить посетителей конференции CITCON, Чикаго 2006 год, за обмен опытом в области CI и проверки. В частности, я очень признателен Полу Джулиусу и Джеффри Фредерику (Jeffrey Frederick) за организацию конференции и всем остальным, участвовавшим в этом событии.

И наконец, я хотел бы поблагодарить Джейн за ее постоянную поддержку и помощь в преодолении трудностей на протяжении создания этой книги.

Поль М. Дюваль, Ферфакс, Вирджиния
Март 2007

Соглашения, принятые в этой книге

При оформлении книги использованы соглашения, общепринятые в компьютерной литературе.

- Новые термины в тексте выделяются *курсивом*. Чтобы обратить внимание читателя на отдельные фрагменты текста, также применяется *курсив*.
- Текст программ, функций, переменных, URL, Web-страниц и другой код представлен моноширинным шрифтом.
- Все, что придется вводить с клавиатуры, выделено **полужирным моноширинным** шрифтом.
- Знакоместо в описаниях синтаксиса выделено *курсивом*. Это указывает на необходимость заменить его фактическим именем переменной, параметром или другим элементом, который должен находиться на этом месте `BINDSIZE=(`*`максимальная ширина колонки`*`)*(`*`номер колонки`*`)`.
- Пункты меню и названия диалоговых окон представлены следующим образом: Menu Option (Пункт меню).
- Разрыв слишком длинных строк кода, не помещающихся на странице, обозначен специальным символом ↳.

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что бы еще вы хотели увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать авторам.

Мы ждем ваших комментариев. Вы можете прислать письмо по электронной почте или просто посетить наш Web-сервер, оставив на нем свои замечания, — одним словом, любым удобным для вас способом дайте нам знать, нравится вам эта книга или нет, а также выскажите свое мнение о том, как сделать наши книги более подходящими для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш e-mail. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию следующих книг. Наши координаты:

E-mail: `info@williamspublishing.com`

WWW: `http://www.williamspublishing.com`

Наши почтовые адреса:

в России: 127055, г. Москва, ул. Лесная, д. 43, стр. 1

в Украине: 03150, Киев, а/я 152