Непрерывная интеграция

улучшение качества программного обеспечения и снижение риска

title.indd 1 02.04.2008 14:05:30

Continuous Integration

Improving Software Quality and Reducing Risk

Paul M. Duvall
with
Steve Matyas and Andrew Glover

ADDISON-WESLEY

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokio • Singapore • Mexico City

title.indd 2 02.04.2008 14:05:31

Непрерывная интеграция

улучшение качества программного обеспечения и снижение риска

Поль М. Дюваль,
а также
Стивен Матиас и Эндрю Гловер

Москва • Санкт-Петербург • Киев
2008

title.indd 3 02.04.2008 14:05:31

ББК 32.973.26-018.2.75
Д95
УДК 681.3.07

Издательский дом "Вильямс"

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция *В.А. Коваленко*

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Дюваль, Поль М., **Матиас III**, Стивен М., **Гловер**, Эндрю.

Д95 Непрерывная интеграция: улучшение качества программного обеспечения и снижение риска. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2008. — 240 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1408-8 (рус.)

В этой книге рассматриваются 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 итоге развертывание программного обеспечения. Но главное, в ней описано, как непрерывная интеграция способна снизить риски, которые подстерегают при создании приложений. В системе непрерывной интеграции большинство этих процессов автоматизировано, и они запускаются после каждого изменения разрабатываемого программного обеспечения.

В книге обсуждаются аспекты автоматизации непрерывной интеграции, большинство предоставляемых ей преимуществ в области повторяемых и склонных к ошибкам процессов. Ныне существует множество великолепных инструментальных средств, поддерживающих непрерывную интеграцию как автоматизированный процесс, использующий сервер CI для автоматизации действий. Тем не менее ручной подход к интеграции (при автоматизированной компиляции) вполне может хорошо сработать.

ББК 32.973.26-018.2.75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства Addison-Wesley Publishing Company, Inc.

Authorized translation from the English language edition published by Addison-Wesley Publishing Company, Inc, Copyright © 2007

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise.

No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher.

Russian language edition published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2008

ISBN 978-5-8459-1408-8 (рус.)
ISBN 0-321-33638-0 (англ.)

© Издательский дом "Вильямс", 2008
© Copyright © 2007 Pearson Education, Inc., 2007

Оглавление

Содержание

Об авторах

Поль М. Дюваль (Paul M. Duvall) — главный технический директор консалтинговой фирмы Stelligent Incorporated, а также интеллектуальный лидер групп помощи по быстрой и надежной разработке программного обеспечения, его усовершенствования и оптимизации процесса создания. Он побывал в роли практически каждого участника проекта программного обеспечения, от разработчика и испытателя архитектуры до руководителя проекта. Поль консультировал клиентов в различных отраслях, включая финансы, недвижимость, правительство, здравоохранение и множество независимых производителей. Его часто приглашают на многие известные конференции по программному обеспечению в качестве почетного председателя. Поль написал книгу *Automation for the People* из серии *IBM developerWorks*, а также является соавтором книг *NFJS 2007 Anthology* (издательство Pragmatic Programmers, 2007) и *UML 2 Toolkit* (издательство Wiley, 2003). Еще он соавтор системы управления данными клинических исследований, а также метода устойчивости к задержкам. Его блоги доступны на сайтах `www.testearly.com` и `www.integratebutton.com`.

Стивен М. Матиас III (Stephen M. Matyas III) — вице-президент компании AutomateIT, отделения службы 5AM Solutions, Inc., которая помогает организациям в улучшении разработки программного обеспечения за счет автоматизации. Стив имеет обширный опыт по созданию прикладного программного обеспечения, включая работу как с коммерческими, так и с правительственными заказчиками. Стив занимал множество различных должностей, от бизнес-аналитика и руководителя проекта до разработчика, дизайнера и архитектора. Еще он является соавтором книги *UML 2 Toolkit* (издательство Wiley, 2003). Стивен на практике применяет многие из итерационных и инкрементных методов разработки, включая Agile и Rational Unified Process (RUP). Он обладает огромным профессиональным опытом в области разработки специального программного обеспечения с использованием Java/J2EE, причем специализируется на методиках, качестве программного обеспечения и усовершенствовании процесса разработки. Стивену присвоена степень бакалавра наук по информатике политехнического института штата Вирджиния, а также государственного университета.

Эндрю Гловер (Andrew Glover) — президент консалтинговой фирмы Stelligent Incorporated, интеллектуальный лидер групп помощи по быстрой и надежной разработке программного обеспечения, его улучшения и оптимизации процесса создания. Энди часто выступает на различных конференциях, а также на симпозиуме No Fluff Just Stuff Software Symposium; кроме того, он соавтор книг *Groovy in Action* (издательство Manning, 2007), *Java Testing Patterns* (издательство Wiley, 2004) и *NFJS 2006 Anthology* (серия *Pragmatic Programmers*, 2006). Он также автор различных сетевых публикаций, включая такие порталы, как developerWorks от IBM, ONJava и Dev2Dev. Его блоги о качестве программного обеспечения доступны на сайтах `www.thediscoblog.com` и `www.testearly.com`.

О соавторах

Лайза Портер (Lisa Porter) — старший технический разработчик для консалтинговых групп, предоставляющих решения по сетевой безопасности для правительства США. До этой книги она занималась техническим редактированием. На ее счету участие в многочисленных больших проектах по разработке программного обеспечения, позволивших ей приобрести богатый опыт в определении требований, а также возможностей реализации проекта. Она также применила принципы технического редактирования в области перевода на другие языки, проектирования и архитектуры. Лайза редактирует книги и сетевые публикации начиная с 2002 года.

Эрик Тавела (Eric Tavela) — главный архитектор компании по разработке программного обеспечения 5AM Solutions, Inc., специализирующейся на внедрении новейших технологий, а также поддержке научных исследований. Основной специализацией Эрика является проектирование и реализация приложений Java/J2EE, а также обучение разработчиков методам объектно-ориентированного программирования и моделирования UML.

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 бо́льший, 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.

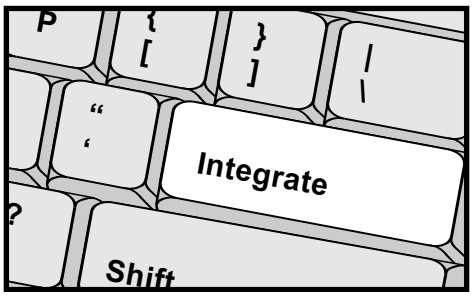

Рис. П.1 Кнопка <Integrate>!

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).