Введение

За последние несколько лет в индустрии информационной технологии (ИТ) стремительно распространились эволюционные методологии разработки программного обеспечения, часто называемые адаптивными, такие как экстремальное программирование (Extreme Programming — XP), метод Scrum, унифицированный процесс компании Rational (Rational Unified Process — RUP), адаптивный унифицированный процесс (Agile Unified Process — AUP) и разработка на основе функций (Feature-Driven Development — FDD). Чтобы было проще понять особенности этих методов, подчеркнем, что эволюционный метод по своему характеру является и итеративным, и инкрементным, а адаптивный подход является эволюционным и вместе с тем характеризуется высокой степенью взаимодействия участников разработки. Кроме того, в организациях, применяющих информационные технологии, все шире внедряются такие адаптивные методики, как рефакторинг, программирование в паре, разработка на основе тестирования (Test-Driven Development — TDD) и адаптивное проектирование на основе модели (Agile Model-Driven Development — AMDD). Эти подходы и методики были разработаны и получили свое развитие в течение многих лет в среде рядовых разработчиков и доведены до совершенства обычными программистами, а не придуманы теоретиками, живущими в башнях из слоновой кости. Короче говоря, эти эволюционные и адаптивные методики, по-видимому, невероятно успешно действуют на практике.

В своей оригинальной книге *Refactoring* Мартин Фаулер дал определение *рефакторинга* как небольшого изменения в исходном коде, которое способствует улучшению проекта кода без изменения его семантики. Иными словами, рефакторинг — это улучшение качества сделанной вами работы без нарушения или добавления чего-либо. Кроме того, в своей книге Мартин обсуждает идею, что если возможно подвергнуть рефакторингу прикладной исходный код, то есть возможность подвергнуть рефакторингу схему базы данных. Но Мартин указал, что рефакторинг баз данных — очень сложная задача, поскольку базы данных отличаются высокой степенью связности; поэтому он решил исключить эту тематику из своей книги.

После публикации книги *Refactoring* в 1999 году оба автора настоящей книги стали искать способы проведения рефакторинга схем базы данных. Первоначально мы работали отдельно, встречаясь друг с другом на таких конференциях, как *Software Development* (`www.sdexpo.com`), и ведя переписку с помощью списков рассылки (например, `www.agiledata.org/feedback.html`). Мы обсуждали идеи, посещали лекции и презентации друг друга на конференциях и вскоре обнаружили, что наши идеи и методы пересекаются, а также являются весьма взаимосовместимыми. Поэтому мы объединили свои усилия в написании настоящей книги, чтобы поделиться своим опытом и рассказать о методах развития схем баз данных путем проведения операций рефакторинга.

Все примеры, приведенные в книге, написаны на языке Java и на языках баз данных Oracle. Практически каждое описание операции рефакторинга базы данных включает код, предназначенный для модификации непосредственно самой схемы базы данных, а

применительно к некоторым наиболее интересным операциям рефакторинга мы показали, какое влияние они могут оказать на прикладной код Java. Безусловно, нельзя найти две базы данных, которые были бы неотличимыми друг от друга, поэтому мы включили описания альтернативных стратегий реализации в тех случаях, когда между программными продуктами баз данных имеются тонкие, но важные различия. В некоторых случаях мы обсуждаем альтернативные реализации каких-то аспектов рефакторинга с помощью таких характерных для СУБД Oracle средств, как команды `SET UNUSED` или `RENAME TO`, а во многих приведенных нами примерах кода используются средства `COMMENT ON`, предусмотренное в СУБД Oracle. В других программных продуктах баз данных предусмотрены другие средства, позволяющие упростить рефакторинг баз данных, и хороший администратор базы данных должен знать, как воспользоваться этими особенностями в своих интересах. Но лучше всего, если в будущем появятся инструментальные средства рефакторинга баз данных, которые могли бы выполнять всю эту работу за нас. Кроме того, мы стремились показать настолько простой код Java, чтобы можно было преобразовать его в код C#, C++ или даже Visual Basic практически без затруднений.

Необходимость в осуществлении эволюционной разработки баз данных

Настало такое время, что эволюционные подходы к разработке баз данных должны выйти на передний план. Это означает, что схема базы данных не должна полностью проектироваться заранее еще до начала проекта; вместо этого схема наращивается на протяжении всего периода осуществления проекта, отражая изменения в требованиях, которые выдвигают лица, заинтересованные в разработке проекта. Нравится нам это или нет, но по мере развития проекта требования изменяются. В традиционных подходах принято отрицать эту фундаментальную истину и пытаться "управлять изменениями"; термин *управление изменениями*, кроме прочих отрицательных оттенков, несет смысл — создание препятствий для изменений. Вместо этого специалисты, непосредственно использующие современные методы разработки, решили принять потребность в изменениях как должное и взять на вооружение методы, позволяющие им совершенствовать плоды своего труда настолько оперативно, насколько это согласуется с расширяющимися требованиями. Программисты стали руководствоваться такими методами, как TDD, рефакторинг и AMDD, а также создали новые инструментальные средства разработки, чтобы упростить использование этих методов. Достигнув этой цели, специалисты в области программного обеспечения осознали, что нужны также методы и инструментальные средства для поддержки эволюционной разработки баз данных.

Преимущества эволюционного подхода к разработке баз данных включают следующее.

1. **Минимизация бесполезных затрат.** Эволюционный, своевременный (Just-In-Time — JIT) подход позволяет исключить издержки, которые неизбежно возникают при использовании последовательных методов в случае изменения требований. Все инвестиции, заблаговременно вложенные в подготовку детализированных требований, создание архитектуры и проектирование артефактов, становятся напрасно потерянными, если в дальнейшем обнаруживается, что требование, ради выполнения которого были сделаны эти затраты, больше не выдвигается. Ведь если ваша квалификация позволяет сделать работу заранее, то,

очевидно, вы сможете воспользоваться своей квалификацией, чтобы сделать ту же работу, когда для этого настанет время.

2. **Предотвращение необходимости в существенных переделках.** Как будет показано в главе 1 "Эволюционная разработка базы данных", все равно должно быть заранее проведено некоторое заблаговременное начальное проектирование, позволяющее продумать наперед основные проблемы и выяснить, какие сложности могли бы потенциально привести к существенным переработкам при их обнаружении на более поздних этапах проекта; не нужно лишь предварительно изучать мельчайшие подробности.
3. **Постоянная уверенность в наличии работоспособной системы.** Эволюционный подход позволяет регулярно выпускать рабочее программное обеспечение (даже если его развертывание осуществляется только в демонстрационной среде), которое всегда может быть передано в эксплуатацию. Если в вашем распоряжении каждый раз через одну-две недели оказывается новая, работоспособная версия системы, риск неудачного завершения проекта резко уменьшается.
4. **Постоянная уверенность в том, что существующий на данный момент проект базы данных имеет наивысшее возможное качество.** Именно в этом состоит вся суть подхода, основанного на проведении рефакторинга баз данных: усовершенствование проекта схемы на основе постепенно осуществляемых небольших изменений.
5. **Применение подхода к разработке, совместимого с подходом других разработчиков.** Разработчики программного обеспечения руководствуются эволюционным подходом, и если специалисты в области обработки данных хотят стать равноправными членами современных групп разработчиков, они также должны выбрать для своей работы один из эволюционных методов.
6. **Сокращение общей трудоемкости.** Применяя в своей производственной деятельности эволюционный подход, вы выполняете только ту работу, которая фактически нужна сегодня, и ни на йоту больше.

Тем не менее при проведении разработки базы данных на основе эволюционных методов необходимо учитывать некоторые возникающие при этом сложности, которые описаны ниже.

1. **Наличие разных подходов у различных категорий разработчиков.** Многие специалисты в области обработки данных предпочитают придерживаться последовательного подхода к разработке программного обеспечения и часто продолжают утверждать, что до начала программирования должны быть созданы в той или иной форме детализированные логические и физические модели данных, которые должны быть взяты за основу. После перехода к использованию современных методологий приходится отказываться от этого подхода, который теперь рассматривается как слишком неэффективный и рискованный; в связи с этим многие специалисты в области обработки данных чувствуют себя растерянными. Но хуже всего то, что многие "идейные руководители" в сообществе специалистов по базам данных — это люди, чье профессиональное становление произошло в 1970-х и 1980-х годах, но для них осталась незамеченной объектная революция, происшедшая в 1990-х годах, поэтому они не

смогли своевременно приобрести опыт в эволюционных разработках. Мир изменился, но эта категория людей, по-видимому, не хочет изменяться вместе с ним. Как описано в данной книге, специалисты в области обработки данных не только имеют возможность организовать свою работу по-новому, на основе эволюционных или даже адаптивных методов, но и сами эти методы фактически являются предпочтительным способом организации работы.

2. **Низкая скорость обучения.** Для изучения описанных в данной книге методов потребуется время, но еще больше времени займет полная перестройка сложившихся последовательных подходов и переход к использованию эволюционных подходов.
3. **Отсутствие полностью сложившихся инструментальных средств поддержки.** Ко времени публикации книги *Refactoring* в 1999 году отсутствовали какие-либо инструментальные средства, поддерживающие эту методику. Но всего лишь через несколько лет в каждой отдельной интегрированной среде разработки (Integrated Development Environment — IDE) появились непосредственно встроенные функции рефакторинга кода. Ко времени написания этой книги также не существовали какие-либо инструментальные средства рефакторинга баз данных, но авторы фактически включили весь код, необходимый для реализации операций рефакторинга вручную. К счастью, в проспекте проектов Eclipse Data Tools Project (DTP) указано на необходимость разработки функциональных средств рефакторинга баз данных в составе интегрированной среды разработки Eclipse, поэтому остается лишь дождаться того, что поставщики инструментальных средств подхватят это начинание.

Суть адаптивных методов

Безусловно, данная книга не посвящена непосредственно адаптивной разработке программного обеспечения, но нельзя отрицать того факта, что основным методом для разработчиков, руководствующихся адаптивным подходом, должен стать рефакторинг баз данных. Процесс рассматривается как адаптивный, если он соответствует четырем критериям, разработанным организацией Agile Alliance (`www.agilealliance.org`). Эти критерии определяют предпочтения, а не альтернативы, побуждая стремление сосредоточиваться на определенных областях, но не исключать другие. Иными словами, если для вас представляют ценность одни сравниваемые концепции, то вы должны еще больше ценить другие концепции, рассматриваемые в сравнении. Например, нельзя отрицать важность процессов и инструментальных средств, но индивидуумы и способы взаимодействия между ними еще важнее. Четыре адаптивных критерия описаны ниже.

1. **Индивидуумы и способы их взаимодействия ВАЖНЕЕ процессов и инструментальных средств.** Наиболее важными аспектами, требующими размышлений, является то, какие люди участвуют в разработках и как они взаимодействуют. (Если вам не удастся добиться успехов в этой сфере, то лучшие инструментальные средства и процессы окажутся бесполезными.)
2. **Работоспособное программное обеспечение ВАЖНЕЕ всеобъемлющей документации.** Основная цель разработки программного обеспечения состоит в создании работоспособного программного обеспечения, которое отвечает требованиям

лиц, заинтересованных в появлении этого программного обеспечения. Сказанное не означает, что документация больше не нужна; документация, подготовленная должным образом, позволяет узнать, как и почему создана система, и определить, как работать с системой.

3. **Сотрудничество с клиентом ВАЖНЕЕ согласования контракта.** Только клиент может сказать вам, чего он хочет. К сожалению, в этом клиенты не очень сильны; вероятнее всего, они не обладают квалификацией, позволяющей точно определить требования к системе, не излагают правильно эти требования с самого начала, а что хуже всего, со значительной вероятностью меняют свои взгляды по поводу будущей системы с течением времени. Безусловно, важно иметь контракт со своими клиентами, но контракт не может заменить эффективное взаимодействие. Преуспевающие специалисты в области информационной технологии тесно сотрудничают со своими клиентами, не жалеют усилий, чтобы узнать, что действительно требуется их клиентам, и вместе с тем постоянно проводят обучение своих клиентов.
4. **Своевременная реакция на изменения ВАЖНЕЕ выполнения плана.** По мере осуществления разработки системы заинтересованные в этом лица начинают осознавать, что хотели бы внести изменения; обнаруживаются также изменения в деловой среде и в основополагающей технологии. В области разработки программного обеспечения изменчивость является повседневной реальностью, а это означает, что весь план проекта и общий подход могут быть эффективными лишь в том случае, если они будут отражать изменения в среде.

Как читать эту книгу

Основная часть книги, включая главы 6–11, состоит из справочного материала, в котором подробно описана каждая операция рефакторинга. В первых пяти главах изложены фундаментальные идеи и методы эволюционной разработки баз данных и, в частности, описаны принципы рефакторинга баз данных. Эти главы необходимо прочитать в следующем порядке.

- Глава 1 “Эволюционная разработка баз данных” содержит краткий обзор основных принципов эволюционной разработки и методов, которые обеспечивают такую разработку. В этой главе приведены краткие сведения о рефакторинге кода, рефакторинге баз данных, регрессионном тестировании баз данных, эволюционном моделировании данных на основе подхода AMDD, управлении конфигурациями информационных активов баз данных, а также обоснована необходимость в применении отдельных специализированных вариантов среды разработки.
- В главе 2 “Рефакторинг баз данных” подробно рассматриваются концепции, лежащие в основе рефакторинга баз данных, и показано, почему осуществление этой задачи может стать сложным на практике. В этой главе рассматривается также пример проведения рефакторинга базы данных как в простой среде с одним приложением, так и в сложной среде с несколькими приложениями.
- Глава 3 “Процесс рефакторинга баз данных” содержит подробное описание этапов, требуемых для проведения рефакторинга схемы базы данных и в простом, и в сложном варианте среды. В случае базы данных с одним приложением обеспечи-

вается гораздо больший контроль над средой, и в результате этого объем работы по проведению рефакторинга схемы намного уменьшается. В среде с несколькими приложениями необходимо предусмотреть переходный период, в течение которого в базе данных параллельно поддерживаются и старая, и новая схема, что позволяет группам разработчиков приложений обновлять и развертывать создаваемый ими код на производстве.

- В главе 4 "Развертывание на производстве" описан процесс, лежащий в основе развертывания операций рефакторинга баз данных на производстве. Такая задача может оказаться особенно затруднительной в среде с несколькими приложениями, поскольку в подобной среде приходится объединять и тестировать изменения, предлагаемые несколькими разными группами разработчиков.
- В главе 5 "Стратегии рефакторинга баз данных" подытожены некоторые из рекомендуемых методов, обнаруженных нами за многие годы, которые относятся к осуществлению рефакторинга схем базы данных. Кроме того, в этой главе изложено несколько идей, которые мы намереваемся опробовать, но еще не имели возможности этого сделать.

Благодарности

Мы хотим поблагодарить указанных ниже людей за их вклад в создание настоящей книги: Дуга Барри (Doug Barry), Гэри Эванса (Gary Evans), Мартина Фаулера (Martin Fowler), Бернарда Гудвина (Bernard Goodwin), Свена Гортса (Sven Gorts), Дэвида Хея (David Hay), Мишель Хаусли (Michelle Housely), Пола Петралиа (Paul Petralia), Майкла Терстона (Michael Thurston) и Майкла Вайздоса (Michael Vizdos).

Кроме того, Прамод хочет выразить свою признательность Срирам Нарайан (Sriram Narayan), Энди Слокаму (Andy Slocum), Ирфан Шаху (Irfan Shah), Нарайяну Раману (Narayan Raman), Анишеку Агарвалу (Anishek Agarwal) и другим своим товарищам по команде, которые постоянно оспаривали его мнение и научили многому, что касается разработки программного обеспечения. Я хочу поблагодарить также Мартина за то, что он научил меня писать, выступать и, в общем, посоветовал выйти за рамки своей деятельности в компании ThoughtWorks; Кента Бека (Kent Beck) за его поддержку; своих коллег из компании ThoughtWorks, которые во многом мне помогают и позволяют чувствовать себя на работе действительно комфортно; своих родителей Джинаппа и Шобху, которые приложили большие усилия по воспитанию меня и Правина, моего брата, и которые с раннего детства следили за тем, как я пишу, и стремились усовершенствовать мой стиль.

Ждем Ваших отзывов!

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 еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо либо просто посетить наш Webсервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

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

Email: `info@williamspublishing.com`

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

Информация для писем:

из России: 115419, Москва, а/я 783

из Украины: 03150, Киев, а/я 152