Данные нормальные формы, как и сам процесс нормализации, упрекают на том основании, что «это просто здравый смысл», и любой компетентный специалист и сам «естественным образом» спроектирует полностью нормализованную БД без необходимости применения теорию зависимостей, да и сама нормализация, как процесс, является вовсе не обязательной. Между тем, на практике оказывается, что идентифицировать и формализовать принципы здравого смысла, а также их применить – весьма трудная задача, а процесс нормализации существенно упрощает работу по составлению запросов и способствует улучшению масштабируемости, поэтому можно сделать вывод, что изучение нормализации является крайне актуальным направлением, необходимым для любого специалиста, работающего в данной сфере.

УДК 621.762.4

Мелихов В.А, Шнитко А.В.

РАЗРАБОТКА ЧЕРЕЗ ТЕСТИРОВАНИЕ И ПОВЕДЕНИЕ

Белорусский национальный технический университет,
г. Минск, Республика Беларусь
Научный руководитель: канд. техн.наук, доцент Дробыш А.А.

Данные методы – особые подходы к разработке, когда сначала пишутся тесты, а потом, на основе этих тестов, пишется непосредственно сам код приложения. Существует множество факторов, которые побуждают использовать именно эти схемы разработки ПО:

– Детали проекта продумываются еще до их реализации, это помогает абстрагироваться от кода и уловить непонятные моменты в ТЗ на самом раннем этапе, что позволяет избежать лишних ошибок и непонимания в ходе работы над проектом.

– Данные методики помогают наладить коммуникацию между разными членами команды: разработчиком, тестировщиком, менеджером и т.д., что позволяет не только сплотить коллектив, но и вовлечь его членов в активный процесс разработки.

– Также эти методы позволяют заметно раньше находить ошибки в коде, а чем раньше пойман баг, тем дешевле будет стоить его исправление, и тем меньше денег потеряет организация.

– Происходит меньше переходов туда и обратно заданий от разработчика к тестировщику, а это значит, что будет сэкономлено огромное количество времени.

– Будет сильно упрощена работа ручных тестировщиков. Регрессионное тестирование – процесс очень трудоемкий. Если все покрыто автотестами, им достаточно просто описать тест-кейсы, а код может написать автоматизатор или разработчик.

Существует две основные разновидности данных подходов:

Разработка через тестирования или TDD – (Test Driven Development) – это техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода по соответствующим стандартам.

Разработка через поведение или BDD – (Behaviour Driven Development) – это разработка, основанная на описании поведения. То есть, есть отдельный человек (или целая группа) который пишет запросы вида "я как пользователь хочу, когда нажали кнопку пуск, тогда показывалось меню как на картинке и происходило то-то", а затем, основываясь на его запросах, создавался код приложения.

Фактически, BDD является расширением TDD-подхода, однако на практике это не так. Они предназначены для разных целей и для их реализации используются разные инструменты. Также в разных командах разработчиков эти понятия могут интерпретировать по-разному, и часто возникает путаница между ними.

В чем разница между TDD и BDD:

TDD это инструмент для юнит-тестирования, то есть для проверки работы отдельных модулей самих по себе. BDD больше подходит для интеграционного (т.е. для проверки, как отдельные модули работают друг с другом) и системного (т.е. для проверки всей системы целиком) тестирования. TDD проверяет работу функций, BDD – пользовательские сценарии. TDD юнит-тесты пишут сами разработчики. BDD требует объедения усилий разных членов команды.

TDD тесты сразу реализуются в коде на формальном языке, а для BDD чаще всего описываются шаги на языке, понятном всем, а не только разработчикам. Из этого вытекают сразу несколько плюсов

для BDD: 1) Тесты BDD легко изменять, поскольку они часто пишутся почти на чистом английском. 2) Такие тесты может писать «product owner» или другие заинтересованные лица, а не только программисты. 3) Результаты выполнения тестов более "человечные", что упрощает работу с ними и их представление перед заказчиком. 4) Тесты не зависят от целевого языка программирования, что значительно упрощает миграцию кода или теста на другой язык программирования или проект.

Какой из этих подходов лучше и взаимозаменяемые ли они.

Существует мнение, что BDD является все тем же TDD, только «в более красивой упаковке», однако это не верное суждение. BDD появился, чтобы сделать команду разработки ближе к бизнес-процессу, организовать активный диалог между менеджерами, разработчиками и тестировщиками на понятном всем языке. При TDD-подходе общение остается в рамках команды разработчиков: тесты нужно писать на формальных языках программирования, тесты и код лежат в одном месте, их достаточно сложно поддерживать, а тестировщик или менеджер почти никогда не будет смотреть, что написали разработчики, однако программисты будут наверняка уверены, что их код будет «чистым» и работать именно так, как должен; будет в допущено в разы меньше ошибок из-за недопонимания в команде, и, как следствие, сэкономлено большое количество времени.

Если изъяснятся совсем простым языком, то TDD – «делать вещи правильно», то есть разрабатывать надежный код. BDD – «делать правильные вещи», то есть получать на выходе именно то, что хочет и ожидает заказчик или пользователь. Следовательно, данные подходы не взаимозаменяемы, это абсолютно разные инструменты для выполнения разных целей.