

WikipediA

Pазработка через приемочные испытания - Acceptance test– driven development

- Source
- Authors

Languages

- Čeština
- Deutsch
- Español
- Français
- Italiano
- Nederlands
- Polski
- Português
- Türkçe
- Norsk
- Svenska
- Dansk
- Suomen kieli
- Magyar
- Română
- Previous article
- Next article

In other projects

Разработка через приемочное тестирование (ATDD) - это методология разработки, основанная на общении между бизнес-клиентами, разработчиками и тестировщиками. ATDD включает в себя многие из тех же практик, что и спецификация на примере (SBE), разработка на основе поведения (BDD), разработка на основе примеров (EDD) и разработка на основе поддержки, также называемая разработкой на основе сюжетного тестирования (SDD). Все эти процессы помогают разработчикам и тестировщикам понять потребности клиентов до внедрения и позволяют клиентам общаться на их собственном языке предметной области.

ATDD тесно связан с разработкой через тестирование (TDD). Он отличается акцентом на сотрудничество между разработчиком, тестировщиком и бизнес-клиентом. ATDD включает приемочное тестирование, но подчеркивает необходимость написания приемочных тестов до того, как разработчики начнут писать код.

СОДЕРЖАНИЕ

Обзор

Создание

Стратегия тестирования

Критерии приемки и испытания

Формат теста
Полный тест
Контрольный осмотр
Другой тестовый пример
Приемочные испытания проекта

См. Также

ссылки

Внешние ссылки

Обзор

Приемочные испытания проводятся с точки зрения пользователя - внешнего вида системы. Они исследуют внешне видимые эффекты, такие как определение правильного вывода системы при конкретном вводе. Приемочные тесты могут проверить, как меняется состояние чего-либо, например, когда заказ меняется с «оплаченного» на «отправленный». Они также могут проверять взаимодействие с интерфейсами других систем, таких как общие базы данных или веб-службы. Как правило, они не зависят от реализации, хотя их автоматизация может и не выполняться.

Создание

Приемочные испытания создаются при анализе требований и перед кодированием. Они могут разрабатываться совместно инициатором запроса (владельцем продукта, бизнес-аналитиком, представителем заказчика и т. Д.), Разработчиком и тестировщиком. Разработчики реализуют систему с помощью приемочных испытаний. Неудачные тесты дают быструю обратную связь о том, что требования не выполняются. Тесты указаны в условиях бизнес-домена. Затем термины образуют универсальный язык, которым пользуются клиенты, разработчики и тестировщики. Тесты и требования взаимосвязаны. Требование, не прошедшее проверку, может быть реализовано неправильно. Тест, который не относится к требованию, является ненужным тестом. Приемочное испытание, которое разрабатывается после начала внедрения, представляет собой новое требование.

Стратегия тестирования

Приемочные испытания являются частью общей стратегии тестирования. Это тесты, проводимые заказчиком, которые демонстрируют бизнес-цели системы. Компонентные тесты - это технические приемочные тесты, разработанные архитектором, которые определяют

поведение больших модулей. Модульные тесты создаются разработчиком для упрощения поддержки кода. Они часто получаются из приемочных и модульных тестов. Межфункциональное тестирование включает тестирование удобства использования, исследовательское тестирование и тестирование свойств (масштабирование и безопасность).

Критерии приемки и испытания

Критерии приемки - это описание того, что будет проверяться тестом. При таком требовании, как «Как пользователь, я хочу получить книгу из библиотеки», критерием принятия может быть «убедиться, что книга отмечена как извлеченная». Приемочный тест для этого требования дает подробную информацию, так что тест может каждый раз выполняться с одинаковым эффектом.

Формат теста

Приемочные испытания обычно имеют такую форму:

Дано (настройка)

Указанное состояние системы

Когда (триггер)

Разработка программного обеспечения

Активность ядер

Процессы • Требования • Дизайн • Инженерное дело • Строительство • Тестирование • Отладка • Развертывание • Обслуживание

Парадигмы и модели

<u>Гибкий</u> • <u>Чистая комната</u> • <u>Инкрементальный</u> • <u>Прототипирование</u> • <u>Спираль</u> • <u>V</u> модель • Водопад

Методологии и рамки

PAC · DevOps · ПАПА · DSDM · FDD ·

IID · Канбан · Lean SD · Меньше · MDD

· MSF · PSP · РАД · RUP · Безопасный ·

Scrum · CEMAT · TSP · Открыть ·

BBEPX · XP

Вспомогательные дисциплины

Управление конфигурацией • Документация • Гарантия качества программного обеспечения • Управление проектом • Пользовательский опыт

Практики

 $\frac{\mathsf{ATDD} \cdot \mathsf{BDD} \cdot \mathsf{CCO} \cdot \mathsf{CI} \cdot \mathsf{CD} \cdot \mathsf{DDD} \cdot \mathsf{\Pi\Pi}}{\cdot \mathsf{SBE} \cdot \mathsf{BCTahbTe} \cdot \mathsf{TDD}}$

Инструменты

Компилятор • Отладчик • Профайлер • Дизайнер графического интерфейса • Моделирование • IDE • Автоматизация сборки • Автоматизация выпуска • Инфраструктура как код • Тестирование

Стандарты и свод знаний

<u>БАБОК · СММІ · Стандарты IEEE · ISO</u> 9001 · Стандарты ISO / IEC · <u>PMBOK ·</u> <u>SWEBOK · ITIL · IREB</u>

Глоссарии

Искусственный интеллект • Информатика • Электротехника и электроника

Контуры

План разработки программного обеспечения

Произошло действие или событие

Тогда (проверка)

Состояние системы изменилось или был произведен вывод

Кроме того, можно добавлять операторы, начинающиеся с AND, в любой из разделов ниже (Учитывая, Когда, Тогда).

Для примера требования шаги могут быть перечислены как:

'Given Book that has not been checked out And User who is registered on the system When User checks out a book

Then Book is marked as checked out

Полный тест

Предыдущие шаги не включают каких-либо конкретных примеров данных, поэтому они добавляются для завершения теста:

Дано:

Книга, которая не была проверена

Книги	
Заголовок	Проверено
Отличная книга	Нет

Пользователь, который зарегистрирован в системе

Пользователи	
Имя	Сэм

Когда:

Пользователь проверяет книгу

Действие кассы				
Пользователь	Сэм	Проверяет	Отличная книга	

Потом:

Книга отмечена как оформленная

Книги		
Заголовок	Проверено	Пользователь
Отличная книга	да	Сэм

Тестовый экзамен

Изучение теста с конкретными данными обычно приводит к множеству вопросов. Для образца это могут быть:

- Что делать, если книга уже оплачена?
- Что делать, если книги не существует?
- Что делать, если пользователь не зарегистрирован в системе?
- Есть ли дата, когда книгу нужно сдать?
- Сколько книг может получить пользователь?

Эти вопросы помогают выявить недостающие или неоднозначные требования. К ожидаемому результату могут быть добавлены дополнительные детали, такие как срок выполнения. Другие приемочные тесты могут проверить, что такие условия, как попытка получить книгу, которая уже была извлечена, вызывают ожидаемую ошибку.

Еще один тестовый пример

Предположим, бизнес-клиент хотел иметь бизнес-правило, согласно которому пользователь может просматривать только одну книгу за раз. Следующий тест продемонстрирует, что:

Сценарий: убедитесь, что бизнес-правило оформления заказа применяется

Дано:

Книга, которая была проверена

Книги		
Заголовок	Проверено	Пользователь
Отличная книга	да	Сэм
Еще одна замечательная книга	Нет	

Пользователи
Имя
Сэм

Когда:

Пользователь проверяет другую книгу

Действие кассы			
Пользователь	Сэм	Проверяет	Еще одна замечательная книга

Потом:

Возникает ошибка

Возникла ошибка
Описание
Нарушение правил кассового дела

Приемочные испытания проекта

Помимо приемочных испытаний требований, приемочные испытания могут использоваться для проекта в целом. Например, если это требование было частью проекта проверки библиотечных книг, приемочные испытания могли бы проводиться для всего проекта. Их часто называют целями SMART . Пример теста: «Когда новая библиотечная система находится в производстве, пользователи смогут регистрировать книги в три раза быстрее, чем сегодня».

Смотрите также

- Concordion
- FitNesse
- Робот Фреймворк
- Датчик (программное обеспечение)
- Огурец (программное обеспечение)

Рекомендации

Внешние ссылки

• Пример фреймворков автоматизации (http://acceptancetestdrivendevelopment.com)

This page is based on the copyrighted Wikipedia article <u>"Acceptance_test%E2%80%93driven_development"</u> (<u>Authors</u>); it is used under the <u>Creative Commons Attribution-ShareAlike 3.0 Unported License</u>. You may redistribute it, verbatim or modified, providing that you comply with the terms of the CC-BY-SA.

abcdef.wiki is not affiliated with the Wikimedia Foundation

Cookie-policy

To contact us: mail to admin@qwerty.wiki

Change privacy settings