Тестирование методом чёрного ящика

Лекция по курсу «Основы тестирования ПО»

Повторение

Определение

Тестирование методом чёрного ящика — это тип тестирования, основанный на использовании исключительно спецификации

При тестировании методом чёрного ящика тестировщик не имеет доступа к исходному тексту программы и не располагает знаниями о её устройстве

Область применимости

Тестирование методом чёрного ящика применимо всегда, когда есть спецификация:

- модульное тестирование
- интеграционное тестирование
- системное тестирование

Преимущества и недостатки

Преимущества

- Тестирование сфокусировано на спецификации, т. е. непосредственно проверяет соответствие ПО требованиям пользователя
- Тестировщику легче найти ошибки в спецификации
- Тестировщик не имеет пристрастного отношения к уже написанному коду

Недостатки

- Невозможно оценить, насколько хорошо протестирована система

Техники тестирования методом чёрного ящика

1. Тестирование на основе классов эквивалентности
2. Тестирование на основе граничных условий
3. Тестирование по таблицам решений
4. Попарное тестирование
5. Тестирование переходов между состояниями
6. Тестирование сценариев использования

Тестирование на основе классов эквивалентности

Основная идея и заложенные предположения

Основная идея

Тестирование на основе классов эквивалентности заключается в разделении множества входных значений на классы эквивалентности с последующим выбором одного тест-кейса из каждого класса

Предполагается, что входные данные из каждого класса эквивалентности обслуживаются единообразно одним и тем же фрагментом кода, поэтому его достаточно проверить в рамках одного текст-кейса

Пример

Спецификация

Политика найма сотрудников на работу:

- если возраст от 0 до 15 лет, не принимаем
- если возраст — 16 или 17 лет, неполный рабочий день
- если возраст от 18 до 54 лет, полный рабочий день
- если возраст от 55 до 99 лет, не принимаем

Тест-кейсы

- 10
- 17
- 35
- 67
- 170
- -3
- abcd (если применимо)
- null (если применимо)

Примеры в основном излагаются по книге Copeland L. A practitioner’s guide to software test design. 2004

Когда эта техника не работает

Не пишите код так!

```
if (age == 0) result = "NO";
if (age == 1) result = "NO";
...
if (age == 15) result = "NO";
if (age == 16) result = "PART-TIME";
if (age == 17) result = "PART-TIME";
if (age == 18) result = "FULL-TIME";
if (age == 19) result = "FULL-TIME";
...
```

Невозможно догадаться о классах эквивалентности данных в такой программе!

Тестирование на основе граничных условий

Основная идея и мотивация к использованию

Основная идея

Техника граничных условий заключается в выявлении границ между классами эквивалентности с последующим выбором тест-кейсов на этих границах, а также выше и ниже этих границ

Данная техника основана на наблюдении, что программы, как правило, чаще всего ломаются на границах классов эквивалентности входных данных

Иногда проблемы с граничными условиями проникают даже в спецификации

Некорректная спецификация

Политика найма сотрудников на работу:

- если возраст от 0 до 16 лет, не принимаем
- если возраст от 16 до 18 лет, неполный рабочий день
- если возраст от 18 до 55 лет, полный рабочий день
- если возраст от 55 до 99 лет, не принимаем

- Некорректной спецификации не может удовлетворять ни одна реализация!
- Сообщите о некорректной спецификации её автору! (владельцу продукта, архитектору и т. п.)

Пример

Спецификация

Политика найма сотрудников на работу:

- если возраст от 0 до 16 лет, не принимаем
- если возраст от 16 до 18 лет, неполный рабочий день
- если возраст от 18 до 55 лет, полный рабочий день
- если возраст от 55 до 99 лет, не принимаем

Правая граница включается в последующий интервал

Тест-кейсы

- -1
- **0**
- 1
- 15
- **16**
- 17
- **18**
- 19
- 54
- **55**
- 56
- 98
- **99**
- 100

Пример

Спецификация

Политика найма сотрудников на работу:

- если возраст от 0 до 16 лет, не принимаем
- если возраст от 16 до 18 лет, неполный рабочий день
- если возраст от 18 до 55 лет, полный рабочий день
- если возраст от 55 до 99 лет, не принимаем

Правая граница включается в последующий интервал

Тест-кейсы (без дублирования внутри интервалов)

- -1
- **0**
- 15
- **16**
- 17
- **18**
- 54
- **55**
- 98
- **99**
- 100

Тестирование по таблицам решений

Основная идея и область применимости техники

Основная идея

Техника тестирования по таблицам решений заключается в составлении таблицы, описывающей все возможные варианты условий и действий, которые необходимо предпринимать в зависимости от этих условий

- Каждая строка таблицы решений становится основой для тест-кейса
- Подход полезен для тестирования многовариантной бизнес-логики

Терминология таблиц решений

- Условие (condition) — спецификация входных данных, относящаяся к одному определённому способу поведения системы
- Действие (action) — спецификация способа поведения системы или получаемого при определённых условиях результата
- Правило (rule) — конкретное соответствие между условием и действием

Пример

Вычисление величины скидки при страховании жизни студента в зависимости от различных факторов

№ правила	**1**	**2**	**3**	**4**
Условия				
женат/замужем	нет	нет	да	да
академич. задолж.	нет	да	нет	да
Действия				
размер скидки, %	25	0	50	40

Здесь таблица решений непосредственно содержит все необходимые тест-кейсы

Замечания к применению

- Условия могут быть небинарными, тогда для каждой строки таблицы создаётся минимум один тест-кейс, удовлетворяющий условиям
- При выборе тест-кейсов возможно соединение данной техники и техники граничных условий
- Количество правил растёт экспоненциально при увеличении количества условий; для сокращения количества тест-кейсов некоторые условия можно группировать (однако это снижает чувствительность набора тестов)

Попарное тестирование

Проблема комбинаций входных данных: пример

Пусть есть кросс-платформенная, кросс-браузерная система, включающая серверную и клиентскую часть

Необходимо проверить её функционирование для

- 3 браузеров
- 3 серверов приложений
- 3 операционных систем на клиенте
- 3 операционных систем на сервере

Итого: $3^4 = 81$ вариант конфигурации

Техника попарного тестирования позволяет уменьшить количество вариантов до 9

Основная идея и область применимости техники

Основная идея

Техника попарного тестирования заключается в составлении комбинаций тестовых данных таким образом, чтобы в наборе тестов проверялись все возможные пары значений всех параметров

Применяется в тех случаях, когда количество комбинаций входных данных, порождающих одинаковый результат, слишком велико, чтобы можно было протестировать их все

Ортогональные массивы

Определение

Ортогональным массивом $L_n(v^k)$ над алфавитом из v букв (где $n = v^2$) называется называется такая матрица A размера $n \times k$, что субматрица из любой пары её столбцов содержит в своих строках каждую из n комбинаций символов

$$L_9(3^4) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 2 & 1 & 2 \\ 1 & 0 & 2 & 2 \\ 1 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \\ 2 & 0 & 1 & 1 \\ 2 & 1 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{pmatrix}$$

Техника попарного тестирования по шагам

1. Определите параметры, которые будут изменяться в тестах
2. Для каждого параметра определите количество вариантов его изменения
3. Найдите в интернете готовый ортогональный массив необходимого размера (или сгенерируйте его сами с помощью подходящей библиотеки)
4. Постройте тест-кейсы, сопоставив значения в ячейках ортогонального массива с параметрами тестов

Тестирование переходов между состояниями

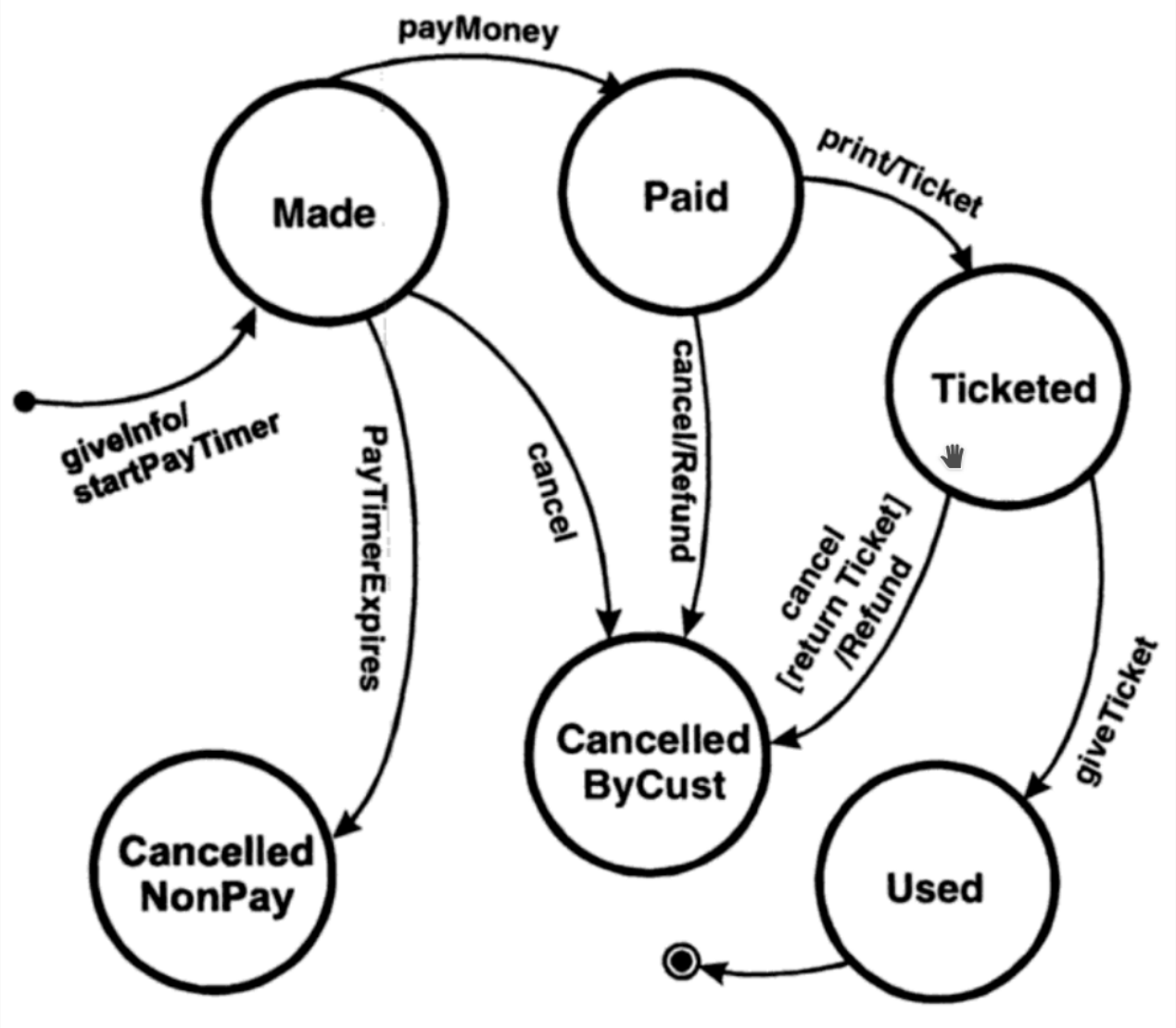
Основная идея и область применимости техники

Основная идея

Техника тестирования переходов между состояниями заключается в выделении состояний системы, определении условий переходов между этими состояниями с дальнейшим покрытием переходов (варианты: вершин, рёбер, путей) графа переходов

Применяется в тех случаях, когда система имеет явно выраженные состояния

Пример системы с состояниями

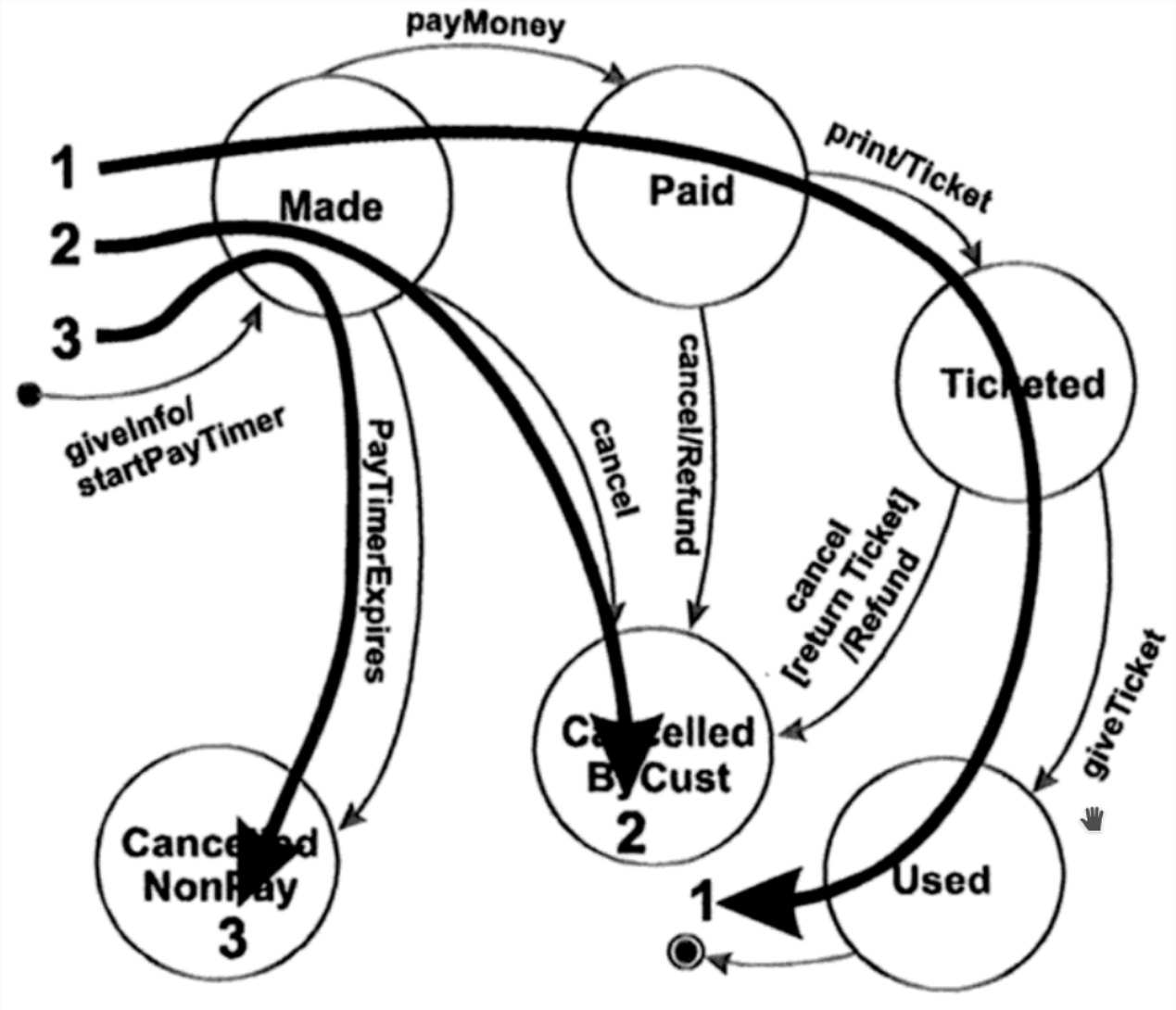


Покрытия графа переходов

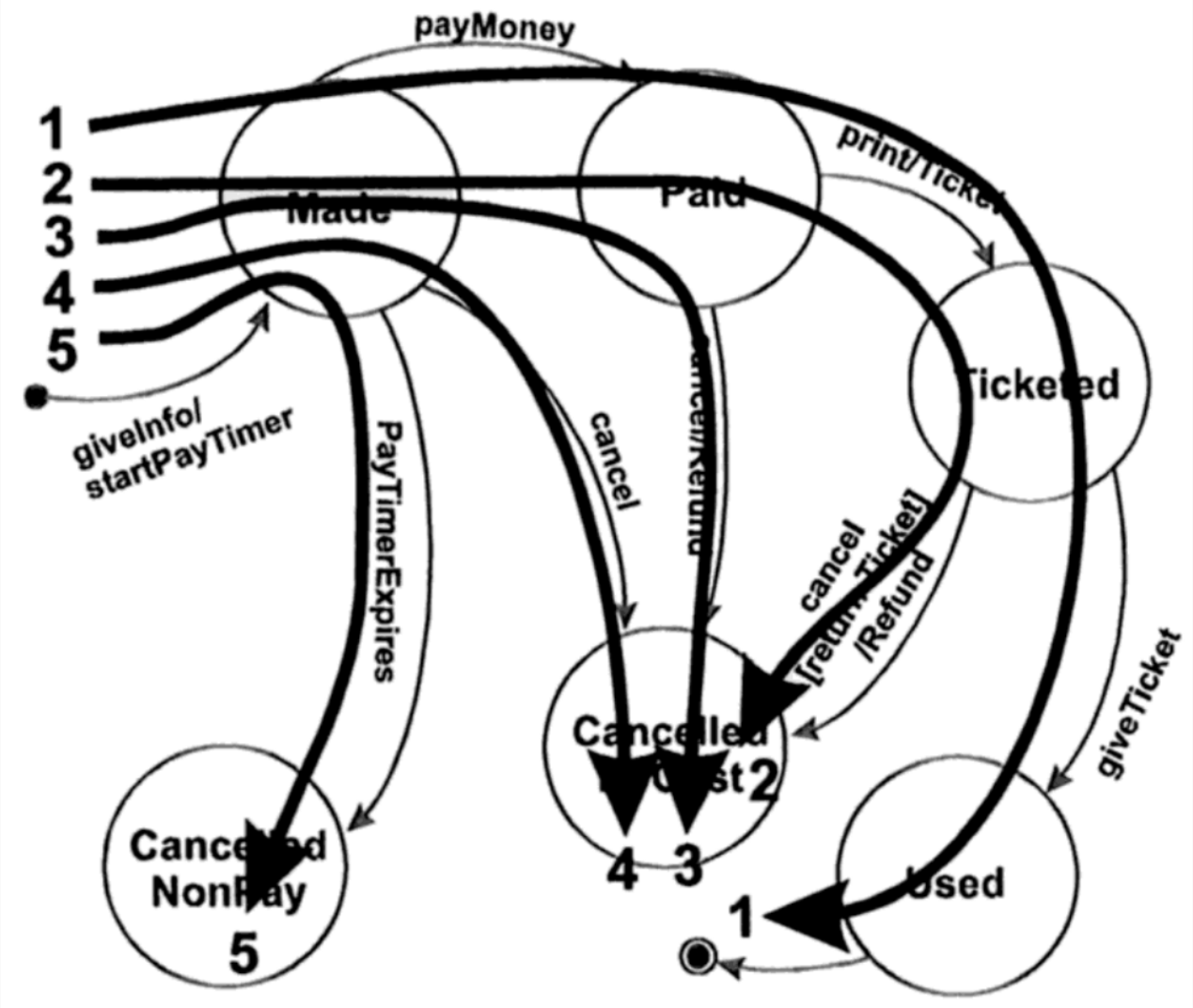
Тест-кейсы строятся на основе одного из покрытий графа переходов:

- покрытие всех состояний (недостаточное)
- покрытие всех событий (недостаточное)
- покрытие всех путей (часто недостижимое)
- покрытие всех переходов (обычно оптимальное)

Покрытие состояний/событий



Покрытие переходов

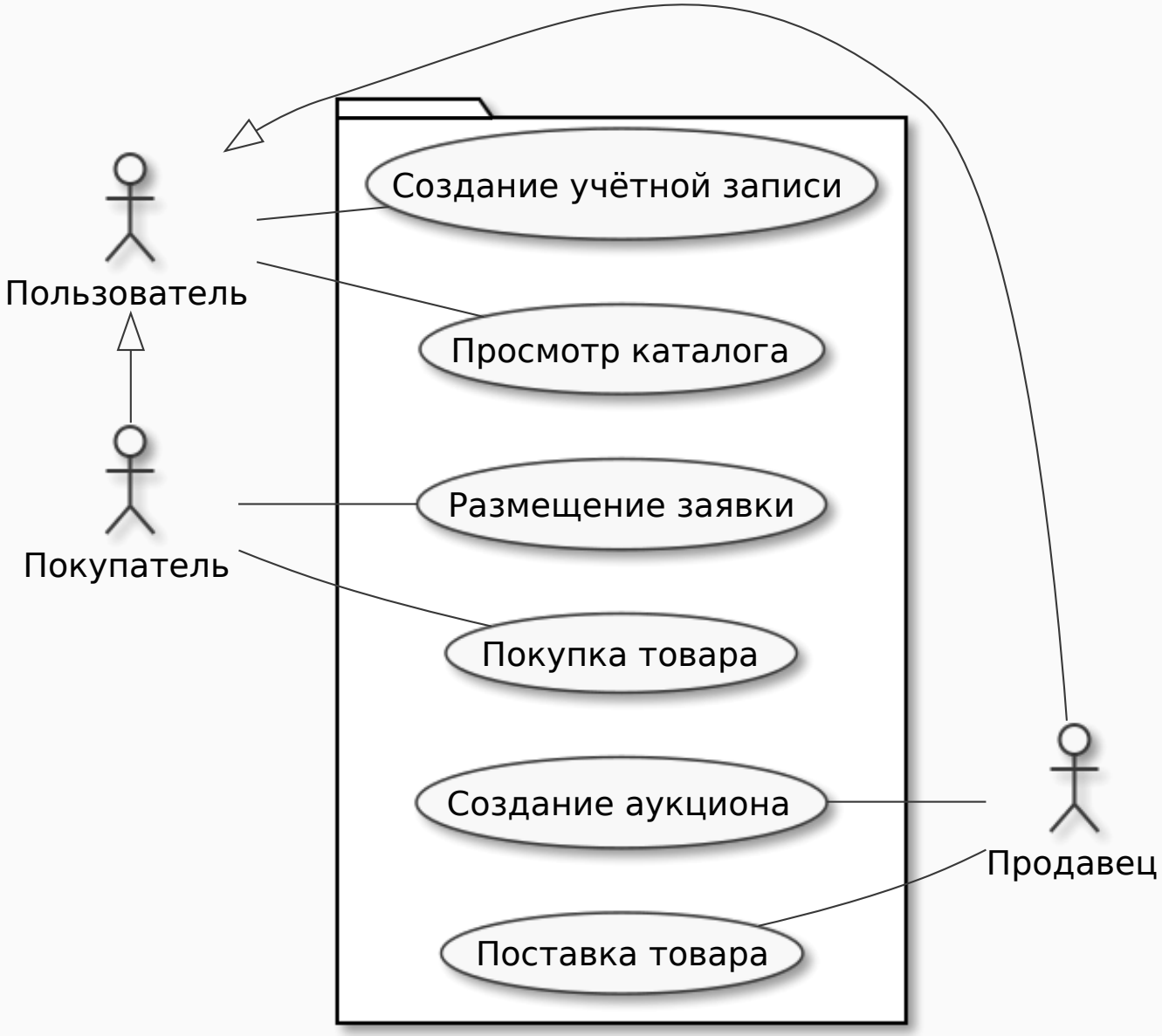


Тестирование сценариев использования

Определение сценариев использования

- Сценарий использования (use case) — это описание типичного способа взаимодействия пользователя с системой и её функционирования в ходе данного взаимодействия
- Актёр (actor) — участник системы, предпринимающий по отношению к ней какие-либо активные действия, движимый некоторой конкретной целью. Термин является неудачным, точнее было бы использовать слово роль
- Транзакция — это выполнение конкретного сценария использования для конкретных данных

Диаграмма сценариев использования



Пример сценария использования (начало)

Use case No.	14
Application	Интернет-магазин
Use case name	Покупка товара
Primary actor	Покупатель
Use case description	Пользователь выбирает товары из каталога с целью их покупки, оплачивает покупку и сообщает данные, необходимые для доставки
Precondition	система не находится в режиме технического обслуживания
Trigger	нет

Пример сценария использования (продолжение)

Basic flow

1. Покупатель просматривает каталог и выбирает товары для покупки
2. Покупатель оценивает стоимость всех товаров
3. Покупатель вводит информацию, необходимую для доставки
4. Система предоставляет полную информацию о цене товара и его доставке
5. Покупатель вводит данные кредитной карточки
6. Система осуществляет авторизацию счёта покупателя
7. Система подтверждает оплату товара
8. Система посылает подтверждение оплаты товара по e-mail

Пример сценария использования (окончание)

Alternate flows

3a Покупатель является постоянным клиентом

.1 Система предоставляет полную информацию о товаре и состоянии счёта заказчика

.2 Покупатель может согласиться или изменить значения по умолчанию, после чего осуществляется переход к шагу 6

6a Система не подтверждает авторизацию счёта

.1 Покупатель может повторить ввод информации кредитной карты или отказаться от покупки

Указания по применению техники

- Необходимо выбрать минимум по одному (обычно нужно намного больше) тест-кейсу на основной поток сценария и каждого из его расширений
- Для выбора входных данных можно использовать техники классов эквивалентности, граничных условий и т. д.
- Поскольку охватить все возможные варианты обычно невозможно, нужно выбирать те из них, которые будут наиболее частыми при эксплуатации системы или сопряжены с наибольшими рисками

Контрольный список для тестирования транзакций

1. Корректные данные, частые транзакции
2. Граничные условия, ошибочные данные
3. Критичные транзакции
4. Все альтернативные потоки
5. Транзакции в разном порядке
6. Нарушенные предусловия
7. Сложные пути, циклы в сценариях
8. Разный порядок ввода данных
9. Странные и глупые вещи