Test Design Techniques

Що будемо вивчати?

- Що таке тест-дизайн
- Які техніки тест-дизайну існують
- Поняття класу еквівалентності та граничних значень
- Правила побудови таблиць прийняття рішень
- Метод парного тестування (Pairwise testing)
- Тестування станів та переходів

Техники тест-дизайну

Техники тест-дизайну – це рекомендації, поради та правила, за якими варто розробляти тести для проведення тестування. Тест дизайн є продумування та написання тестових випадків, відповідно до вимог проекту, критеріями якості майбутнього продукту та фінальними цілями тестування.

Мета тест-дизайну – забезпечити покриття функціоналу докладання тестами те щоб їх було мінімально достатню кількість.

Вирізняють такі техніки тест-дизайну:

- **Еквівалентний поділ** (Equivalence Partitioning);
- **Аналіз граничних значень** (Boundary Value Analysis);
- **Таблиця прийняття рішень** (Decision Table Testing);
- **Тестування станів та переходів** (State-Transition Testing);
- **Метод парного тестування** (Pairwise Testing).

Коли ви вже знаєте, повне тестування навіть простої програми неможливе. А якщо і можливо, то займе багато років.

Візьмемо, наприклад, калькулятор. Скільки комбінацій вхідних параметрів треба протестувати?

Давайте обмежимося тільки операцією складання. Два числа по 8 знаків - це 10^8*10^8 = 10^16 - 10 000 000 000 000 000 комбінацій.
Якщо прикинути, що на кожний тест ми витратимо близько 10 секунд, то час повного тестування займе в нас 10^17 секунд, тобто більше 3 мільярдів років безперервних тестувань.

A-Level UKRAINE

Клас еквівалентності

Клас еквівалентності (equivalence class) – одне чи кілька значень введення, до яких додаток застосовує однакову логіку.

Два тести можна вважати еквівалентними, якщо:

- **Спрямовані на пошук однієї і тієї ж помилки;**
- **Якщо один із тестів виявляє помилку, інші швидше за все, теж її виявлять і навпаки;**
- **Тести використовують подібні набори вхідних даних;**
- **Якщо один із тестів не виявляє помилку, інші, швидше за все, теж її не виявлять;**
- **Для виконання тестів ми здійснюємо одні й самі операції;**
- **Тести генерують однакові вихідні дані або наводять додаток в один і той же стан;**
- **Усі тести призводять до спрацьовування одного й того самого блоку обробки помилок;**
- **Жоден із тестів не призводить до спрацювання блоку обробки помилок.**

Техніка розділення на еквівалентні класи

Поділ на еквівалентні класи – це метод тестування програми, який поділяє вхідні дані програмного модуля на розділи з еквівалентними даними, з яких можна отримати тестові приклади. В принципі, тестові приклади призначені для охоплення кожного розділу хоча б один раз.

Цей метод намагається визначити тестові приклади, які виявляють класи помилок, зменшуючи загальну кількість тестових прикладів, які необхідно розробити. Перевага цього підходу – скорочення часу, необхідного для тестування програмного забезпечення, за рахунок меншої кількості тестових прикладів.

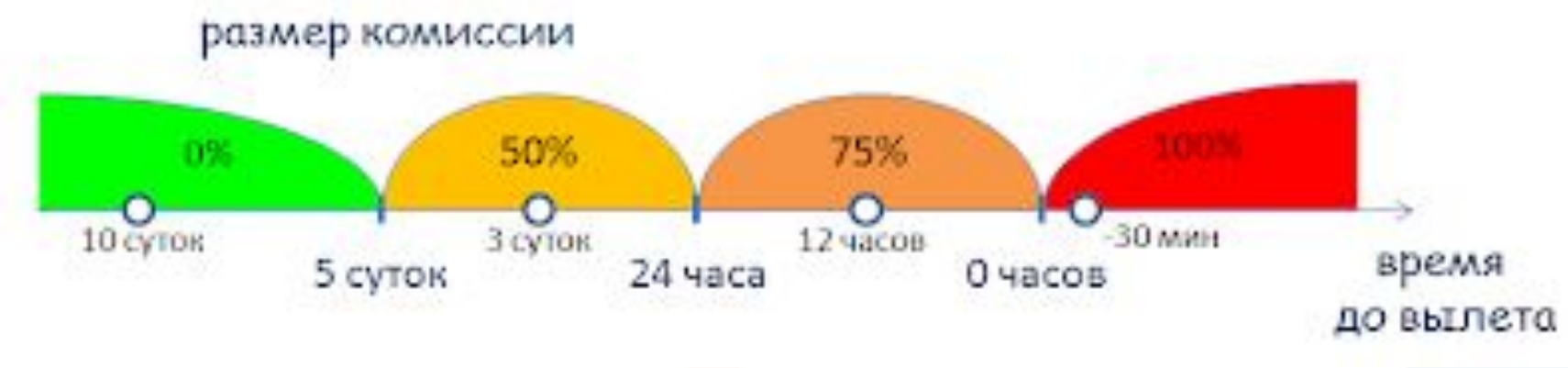
Алгоритм використання техніки

- **Визначити класи еквівалентності.** Це головний крок техніки. Від нього багато в чому залежить ефективність її застосування.
- **Потім потрібно вибрати одного представника кожного класу.** На цьому етапі з кожного еквівалентного набору тестів ми вибираємо один тест.
- **Потрібно виконати випробування.** На цьому кроці ми виконуємо тести кожного класу еквівалентності.

Можна розбивати випробування на класи еквівалентності за різними принципами. Наприклад, якщо ми тестуємо поле введення, яке приймає максимум 5 символів, то можемо вибрати різні принципи розбиття на класи еквівалентності:

- **Кількість символів.**
- **За типом знаків (цифри, літери, спеціалізовані знаки).**

Приклад



Давайте розглянемо функцію підрахунку комісії при скасуванні бронювання авіаквитків. Розмір комісії залежить від часу до вильоту, коли скоєно скасування:

- За 5 діб до вильоту комісія складає 0%;
- Меньше 5 діб, але понад 24 години – 50%;
- Менше 24 години до вильоту – 75%;
- Після вильоту – 100%.

1.ВИЗНАЧИМО КЛАСИ ЕКВІВАЛЕНТНОСТІ:

- 1 клас: час до вильоту > 5 діб.
- 2 клас: 24 години <час до вильоту <5 діб.
- 3 клас: 0 годин <час до вильоту <24 години.
- 4 клас: час до вильоту <0 годин (виліт вже відбувся).

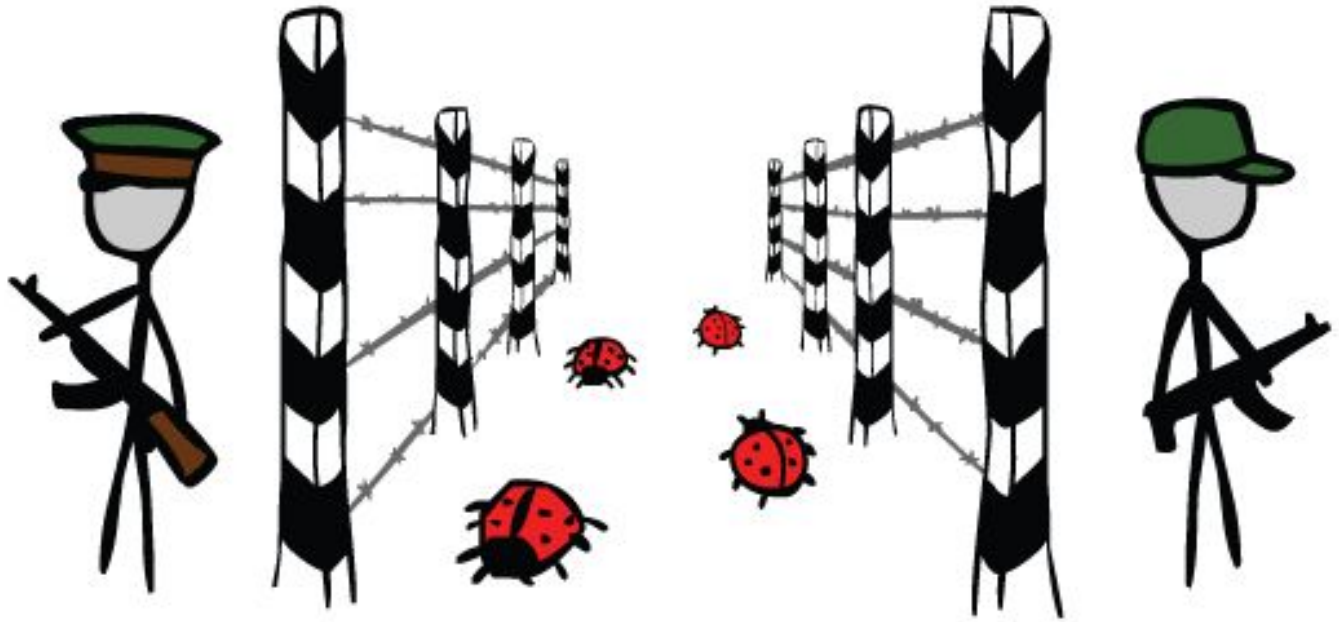
2. ВИБЕРЕМО ОДНОГО ПРЕДСТАВНИКА ВІД КОЖНОГО КЛАСУ:

- Час до вильоту = 10 діб (тест із 1-го класу).
- Час до вильоту = 3 діб (тест із 2-го класу).
- Час до вильоту = 12 годин (тест із 3-го класу).
- Час до вильоту = -30 хв (тест із 4-го класу).

3. ВИКОНАЄМО ТЕСТИ:

- Скасуємо броню за 10 діб до вильоту і перевіримо, що комісія склала 0%.
- Скасуємо броню за 3 доби до вильоту та перевіримо, що комісія склала 50%.
- Скасуємо броню за 12 годин до вильоту і перевіримо, що комісія склала 75%.
- Скасуємо броню через 30 хв після вильоту та перевіримо, що комісія склала 100%.

Баги водяться на межі



A-Level UKRAINE

Техніка аналізу кордонних значень

- Граничні значення – це значення, у яких один клас еквівалентності перетворюється на інший.
- Аналіз граничних значень – це метод тестування програми, який полягає у перевірці поведінки програми у крайніх (граничних) значеннях.
- Граничне тестування також може включати тести, які перевіряють поведінку системи на вхідних даних, що виходять за допустимий діапазон значень. При цьому система має певним способом опрацьовувати такі ситуації. Наприклад, за допомогою виняткової ситуації або повідомлення про помилку.
- Дуже важливо перевіряти саме граничні значення, тому що часто виникають помилки саме на межах класів еквівалентності.

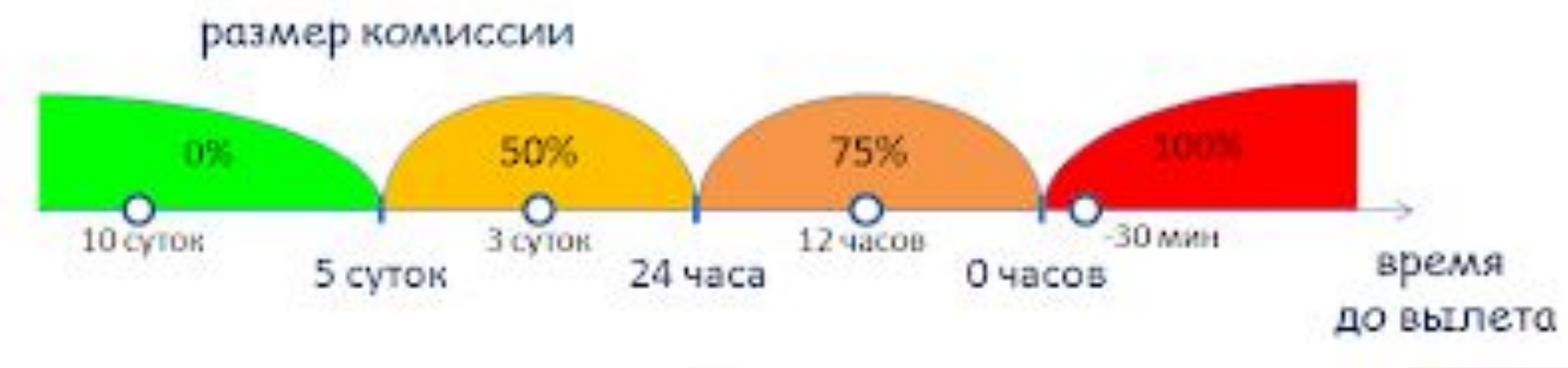
На кожній межі діапазону слід перевірити три значення:

- граничне значення;
- значення перед кордоном;
- значення після кордону.

Алгоритм використання техніки

1. **Визначити класи еквівалентності.**
2. **Потім слід визначити граничні значення цих класів.**
3. **Потрібно розуміти якого класу належить значення.**
4. **Провести тести** з перевірки значення до кордону, на кордоні та одразу після кордону.

Приклад



Давайте розглянемо функцію підрахунку комісії при скасуванні бронювання авіаквитків. Розмір комісії залежить від часу до вильоту, коли скоєно скасування:

- За 5 діб до вильоту комісія складає 0%;
- Меньше 5 діб, але понад 24 години – 50%;
- Менше 24 години до вильоту – 75%;
- Після вильоту – 100%.

1. ВИЗНАЧИМО КЛАСИ ЕКВІВАЛЕНТНОСТІ:

- 1 клас: час до вильоту > 5 діб.
- 2 клас: 24 години <час до вильоту <5 діб.
- 3 клас: 0 годин <час до вильоту <24 години.
- 4 клас: час до вильоту <0 годин (виліт вже відбувся).

2. ВИЗНАЧИМО КОРДОННІ ЗНАЧЕННЯ ЦИХ КЛАСІВ:

Час до вильоту = 5 діб та 1 секунда.

- Час до вильоту = 5 діб
- Час до вильоту = 4 доби 23 години 59 хвилин 59 секунд.
- Час до вильоту = 24 години та 1 секунда
- Час до вильоту = 24 години
- Час до вильоту = 23 години 59 хвилин 59 секунд.
- Час до вильоту = 1 секунда
- Час вильоту

Одна секунда після вильоту

3. ВИЗНАЧИМО ДО ЯКОГО КЛАСУ ВІДНОСИТЬСЯ ЗНАЧЕННЯ.

4. ВИКОНАЄМО ТЕСТИ.

A-Level UKRAINE

Таблиця прийняття рішень

Таблиця прийняття рішень – метод компактного представлення моделі зі складною логікою. Таблиці прийняття рішень, як правило, поділяються на чотири квадрати, як показано нижче:

Умови	**Варіанти виконання умов**
Дії	Необхідність дій

Умови – список можливих умов (виписуємо по горизонталі);

Варіанти виконання умов – комбінація із виконання та/або невиконання умов із цього списку (виписуємо по вертикалі);

Дії - список можливих дій (виписуємо по горизонталі);

Необхідність дій – вказівку треба чи не треба виконувати відповідну дію для кожної з комбінацій умов (виписуємо по вертикалі).

A-Level UKRAINE

Приклад (один результат)

Уявімо, що ми прийшли в страхову компанію для оформлення страховки на автомобіль. Для того, щоб визначити вартість страховки, нам потрібно відповісти на два питання представника страхової компанії:

1. Чи маємо стаж водіння 5 років?
2. Чи ми були винуватцями аварій?

Виходить дві умови по два можливі варіанти, разом чотири варіанти виконання умов (чотири правила). На кожне правило свій результат:

- Якщо у нас невеликий стаж і ми часто буваємо в аваріях – доведеться заплатити по максимуму, інакше страхова компанія невигідно нас страхуватиме;
- Якщо немає стажу, але немає аварій – платимо менше, але не сильно.
- Якщо ми досвідчені водії, але потрапляємо в аварії вкрай рідко і не завжди з нашої вини – цінник ще трохи нижчий.
- Якщо ми досвідчені водії та ще й без аварій – платимо найменше.

Страховка 200 руб!

Страховка 100 руб!

Страховка 50 руб!

Страховка 10 руб!

A-Level
UKRAINE

А тепер те саме, тільки у вигляді таблиці:

	Правило 1	**Правило 2**	**Правило 3**	**Правило 4**
Умови				
Стаж 5 років	Ні	Ні	Так	Так
Участь у аваріях	Так	Ні	Так	Ні
Результат				
Вартість страховки	200	100	50	10

Приклад (множинний результат)

Є інтернет-магазин, який пропонує:

- Знижка постійного покупця;
- Кількість речей, що кур'єр привезе безкоштовно.

Це такі «плюшки» за лояльність та повторну купівлю. Для їх обчислення існують дві умови:

- Скільки покупець витратив грошей в інтернет-магазині – бонус додається при досягненні 100 грн., 500 грн., 1000 грн., 5000 грн.
- Який відсоток викупу – покупець отримує бонус при досягненні 5%, 30%, 50% та 80% викупу.

Спробуємо записати всі умови:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Условия																
Потратил	100	100	100	100	500	500	500	500	1000	1000	1000	1000	1000	1000	1000	1000
Выкуп	5%	30%	50%	80%	5%	30%	50%	80%	5%	30%	50%	80%	5%	30%	50%	80%
Плюшка																
Скидка	0%	5%	5%	7%	5%	6%	7%	8%	8%	8%	10%	15%	11%	13%	20%	20%
Кол-во вещей	2	4	4	6	8	8	9	9	6	10	15	15	15	15	15	20

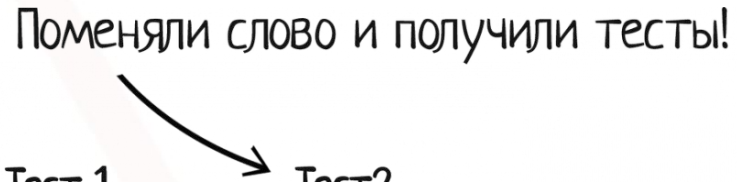
A-Level UKRAINE

Переваги техніки

1. **Наочність – таблиця наочніша за текст.** Можна взяти таблицю та підійти до аналітика з якимось питанням. Або до розробника. Їм буде простіше зрозуміти, про що мова, аніж якщо ви принесете стіну тексту.
2. **Таблиця допомагає глянути на ТЗ свіжим поглядом, по-новому.** Поки ми намагаємося поєднувати умови, складаючи таблицю, ми можемо помітити пропущений раніше баг.
3. **Наочність допоможе знайти баги у документації.** Бо косяк формули відразу кидатиметься в очі.

A-Level UKRAINE

4. Намалював таблицю = записав тест-кейси. Поміняв у заголовках слово "правило" на "тест-кейс", і ось вони, готові випробування! І це будуть основні позитивні тести, які ми проводимо насамперед.



	Тест 1	Тест2		Тест N
	~~Правило 1~~	~~Правило 2~~	...	~~Правило N~~
Условия				
Потратил	100	500	1000	5000
Выкуп	5%	30%	50%	80%
Плюшка				
Скидка	0%	5%	10%	20%
Кол-во вещей	2	4	10	20

Недоліки техніки

Особливих мінусів немає, але таблиця не потрібна, якщо:

1. Занадто проста умова – тому що «але навіщо?». І все зрозуміло.
2. Занадто багато вхідних даних буде дуже багато колонок. Багато тестів, але мало результату, тому що тут уже потрібний тест-аналіз, pairwise і т.д.

A-Level UKRAINE


Таблиця рішень допомагає
подивитись на ТЗ свіжим поглядом

Тестування станів та переходів

Тестування станів та переходів, як і таблиця прийняття рішень, відмінний інструмент для фіксування вимог та опису роботи програми. На відміну від Decision tables testing, які описують конкретний стан програми, State-Transition testing описують, як ці стани програми можуть змінюватися. Діаграми визначають всі події, що виникають під час роботи програми, і як додаток реагує на ці події.

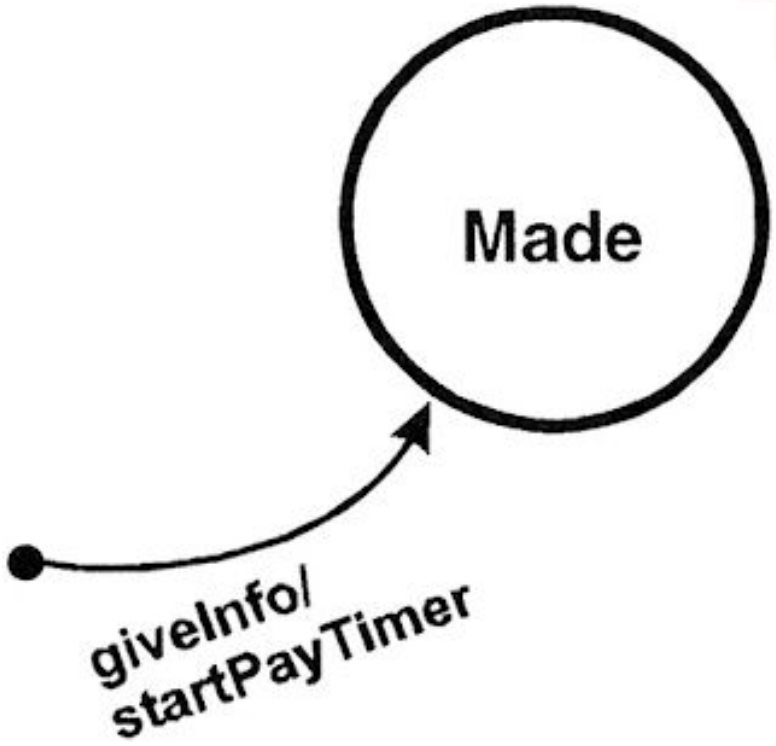
Використання цієї техніки легко пояснити відразу в роботі, на прикладах. Розберемо два види візуального представлення цієї техніки:

- State-Transition Diagrams (діаграми);
- State-Transition Tables (таблиці).

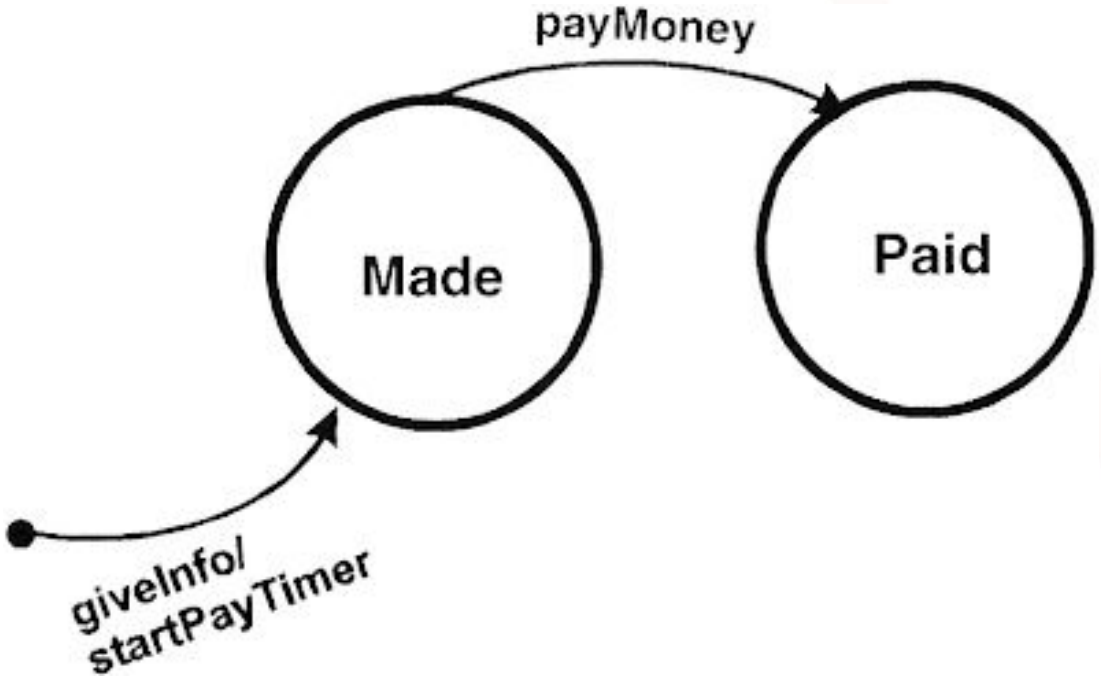
State-Transition Diagrams

Розберемо приклад резервації авіаквитків.

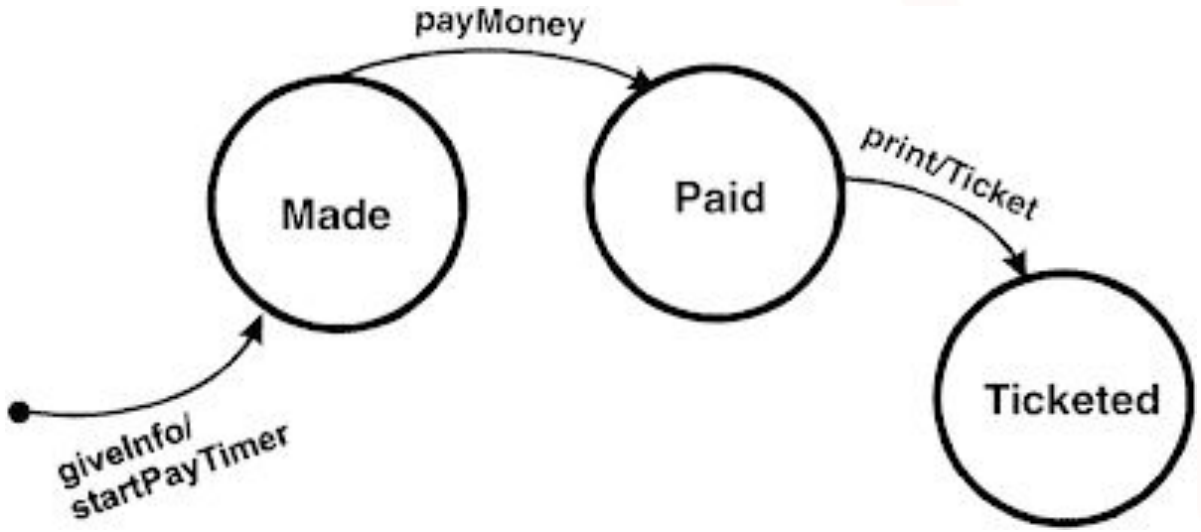
- Спочатку, як клієнти, надаємо авіакомпанії інформацію для резервації – місце відправлення, місце прибуття, дату і час відправлення. Службовець авіакомпанії служить інтерфейсом між нами та системою резервації авіаквитків, і використовує надану нами інформацію для створення резервації.
- Після цього наша резервація перебуватиме у стані "Made" (зроблена).
- До того ж, після створення резервації, система резервації запускає таймер.
- Якщо таймер виходить, а зарезервований квиток не сплачено - система скасовує резервацію.



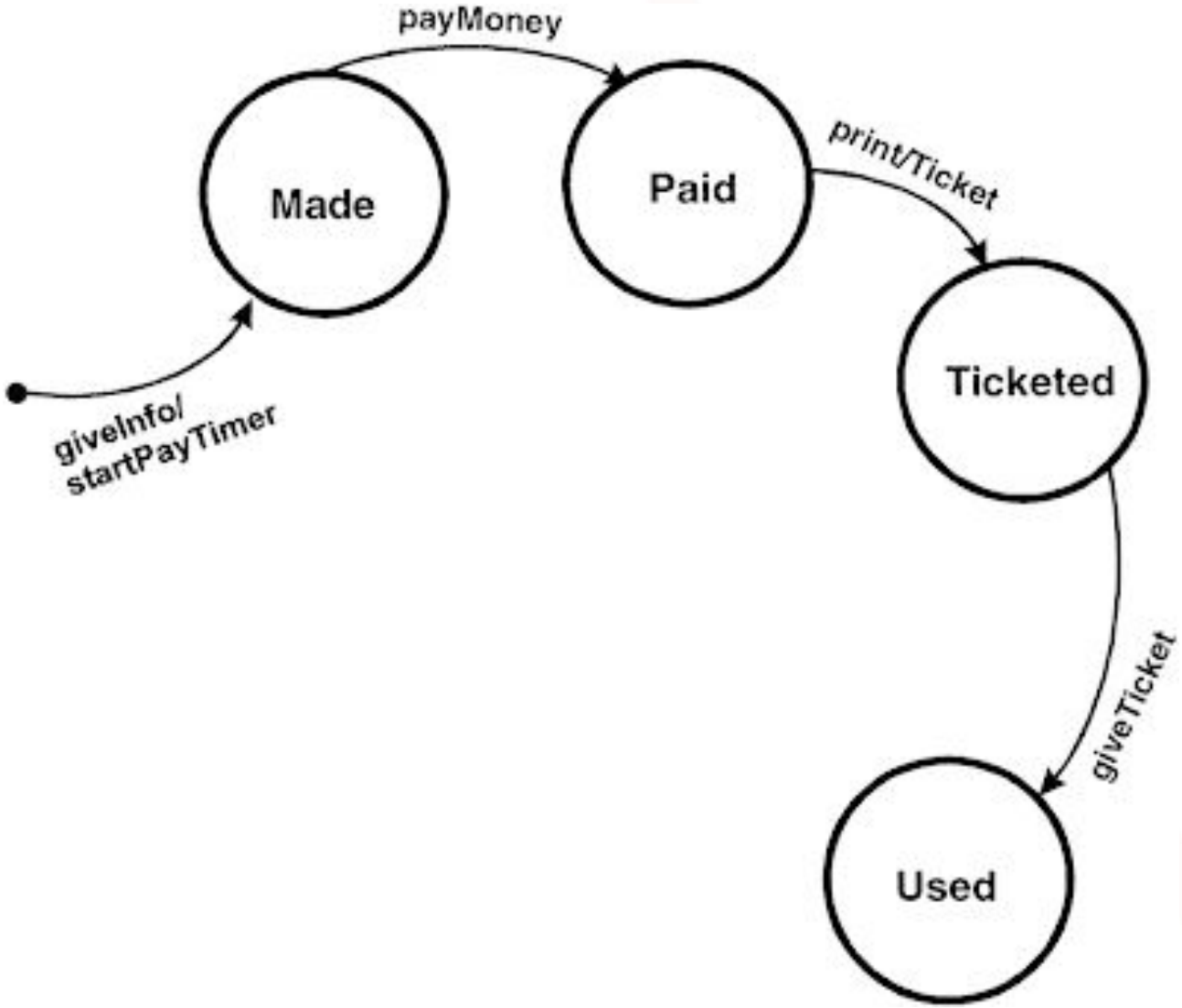
- **Коло** є стан системи резервації авіаквитків, у нашому випадку це "Made" стан.
- **Стрілка** показує перехід у стан "Made".
- **Опис** під стрілкою це подія вихідні ззовні системи (giveInfo).
- **Команда** в описі під стрілкою (після "/") говорить нам, що система зробила якусь дію у відповідь на подію, в нашому випадку це запуск таймера.
- **Чорна точка** вказує на початок/вхід діаграми.



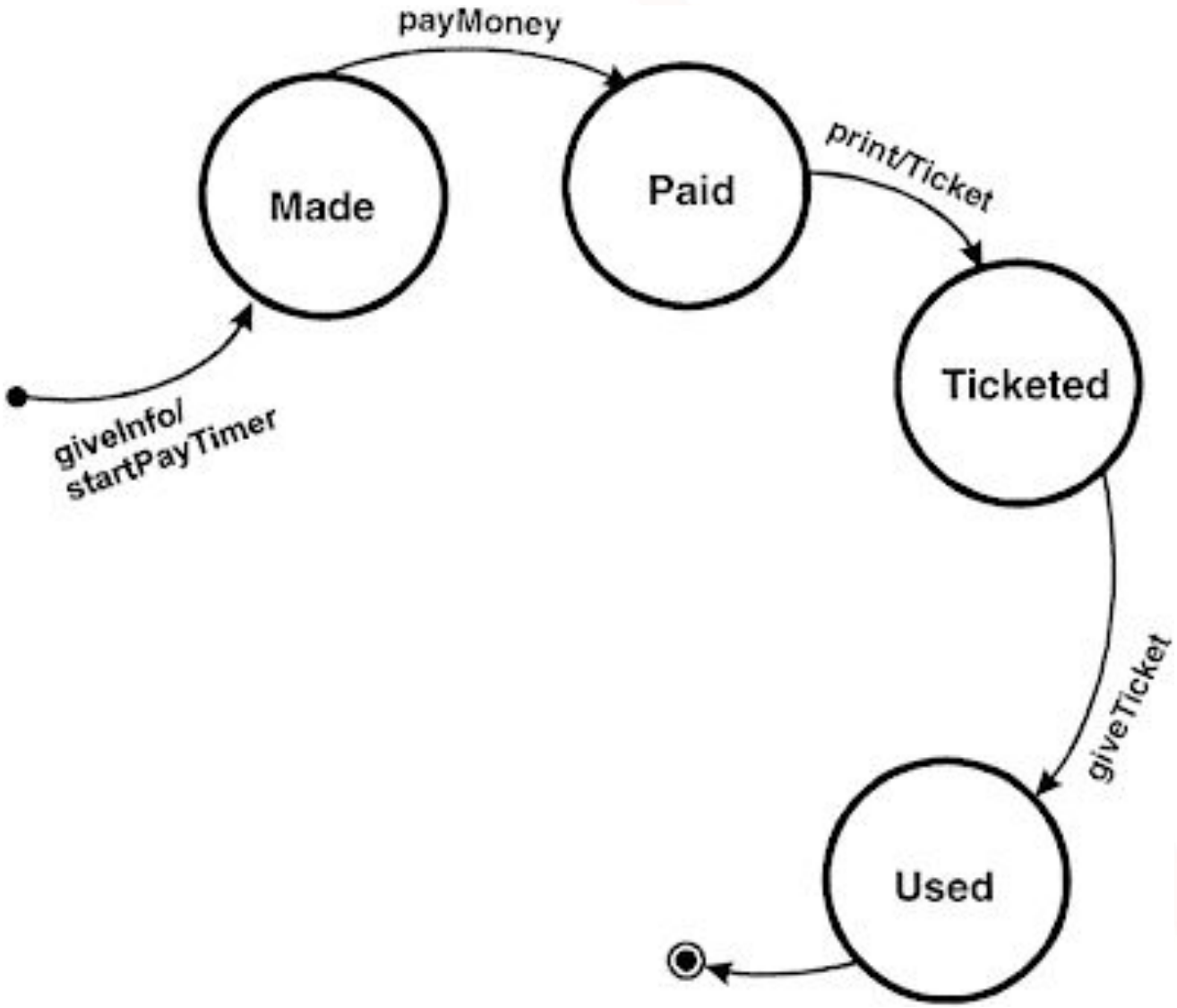
Далі, якщо таймер не спливає і ми оплатили зарезервований квиток, то система набуває стану Paid (оплачений). Це показано стрілкою "payMoney" (заплатити гроші) та переходом зі стану "Made" в стан "Paid".



Зі стану "Paid" повинен бути перехід у стан "Ticketed" (вибілений), коли квиток буде надрукований і переданий нам у руки. Зверніть увагу, що при переході в стан "Ticketed" авіаквиток (Ticket) є вхідними даними стану.



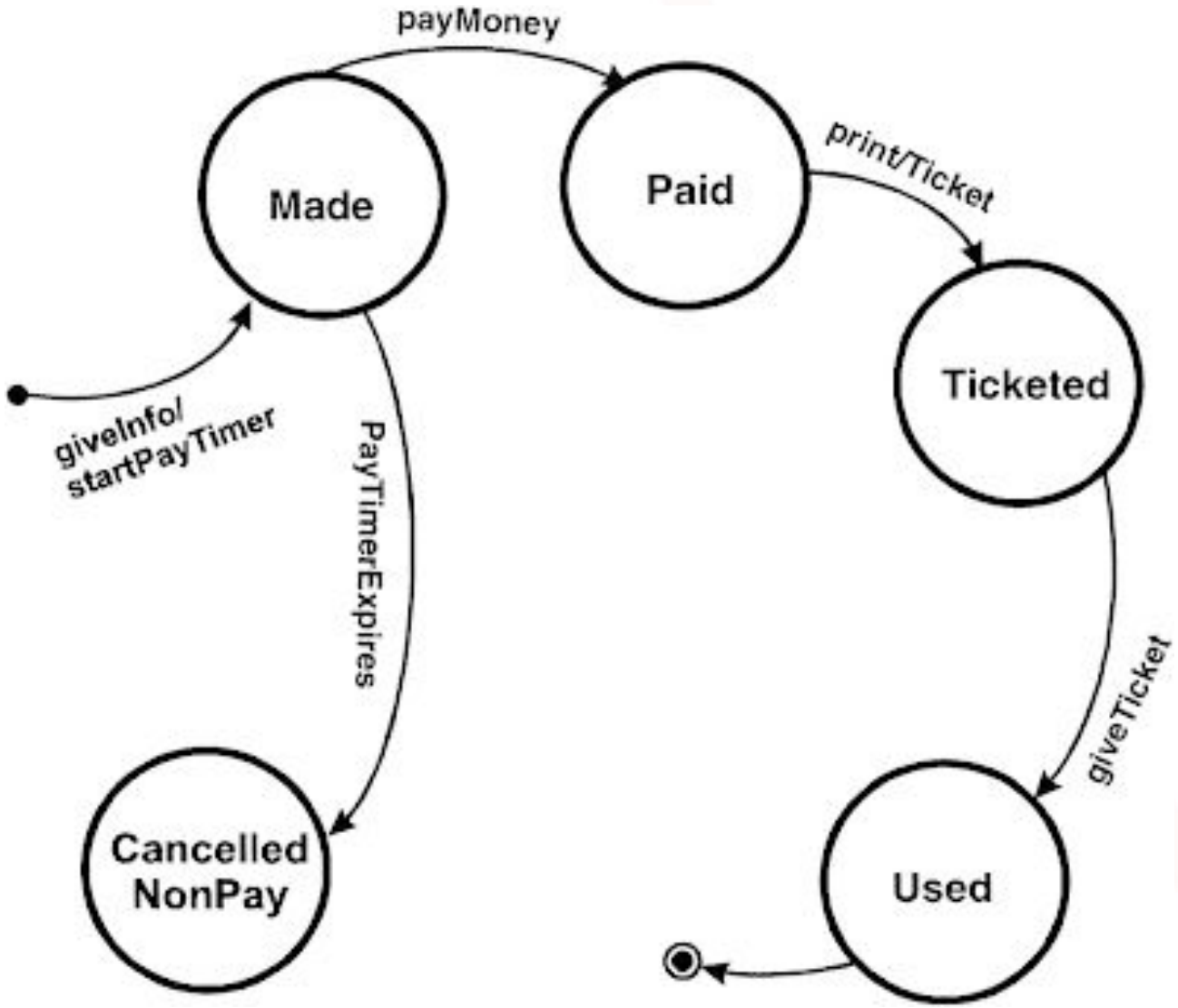
Зі стану "Ticketed" ми переходимо в стан "Used" (використаний), коли віддаємо свій квиток персоналу аеропорту, при посадці в літак.



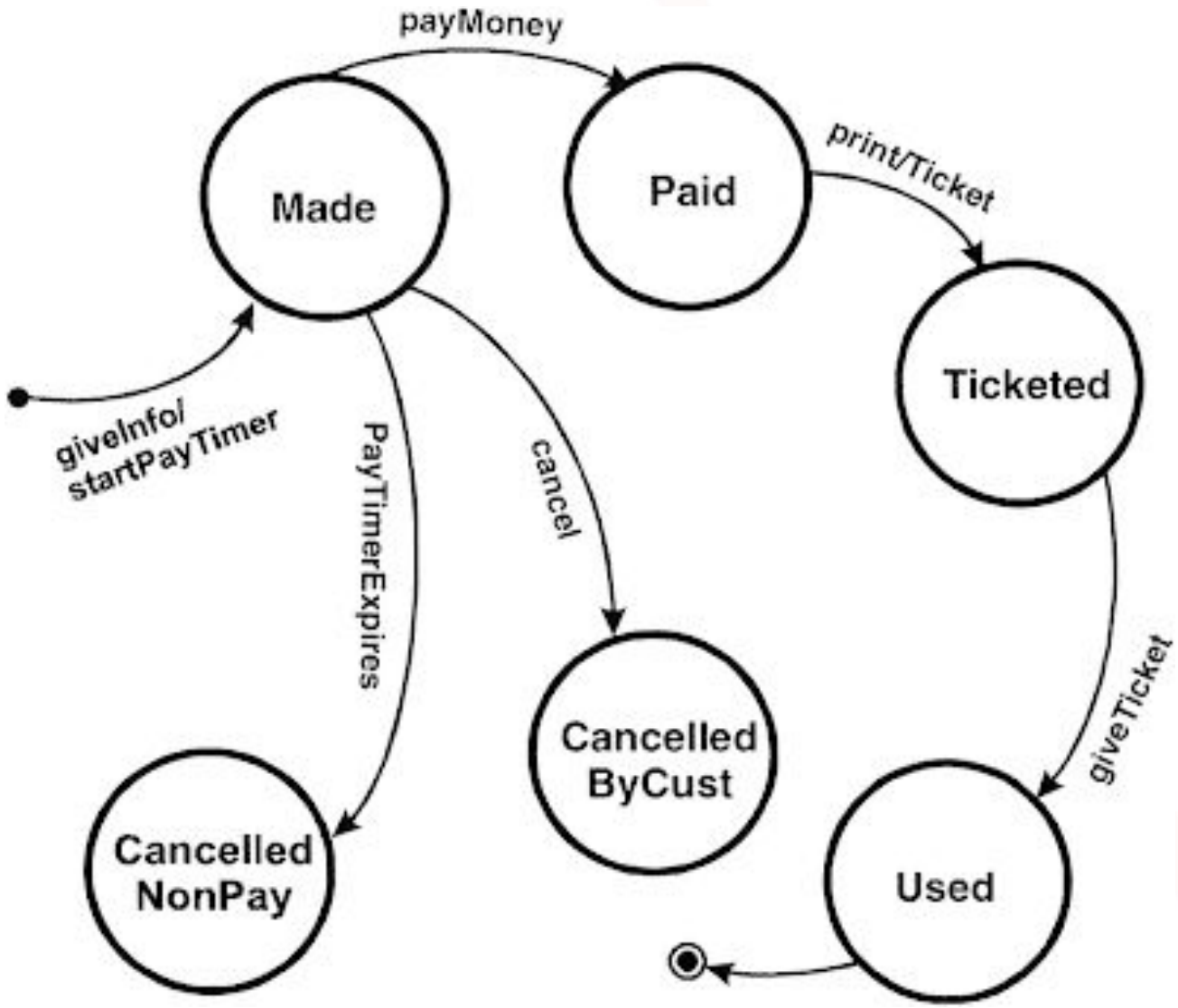
Після якоїсь дії (на цей раз не вказаної на діаграмі) шлях діаграми закінчується символом мішені.

Але ця діаграма показує ще не всі можливі стани та переходи у життєвому циклі резервації. Почнемо доповнювати.

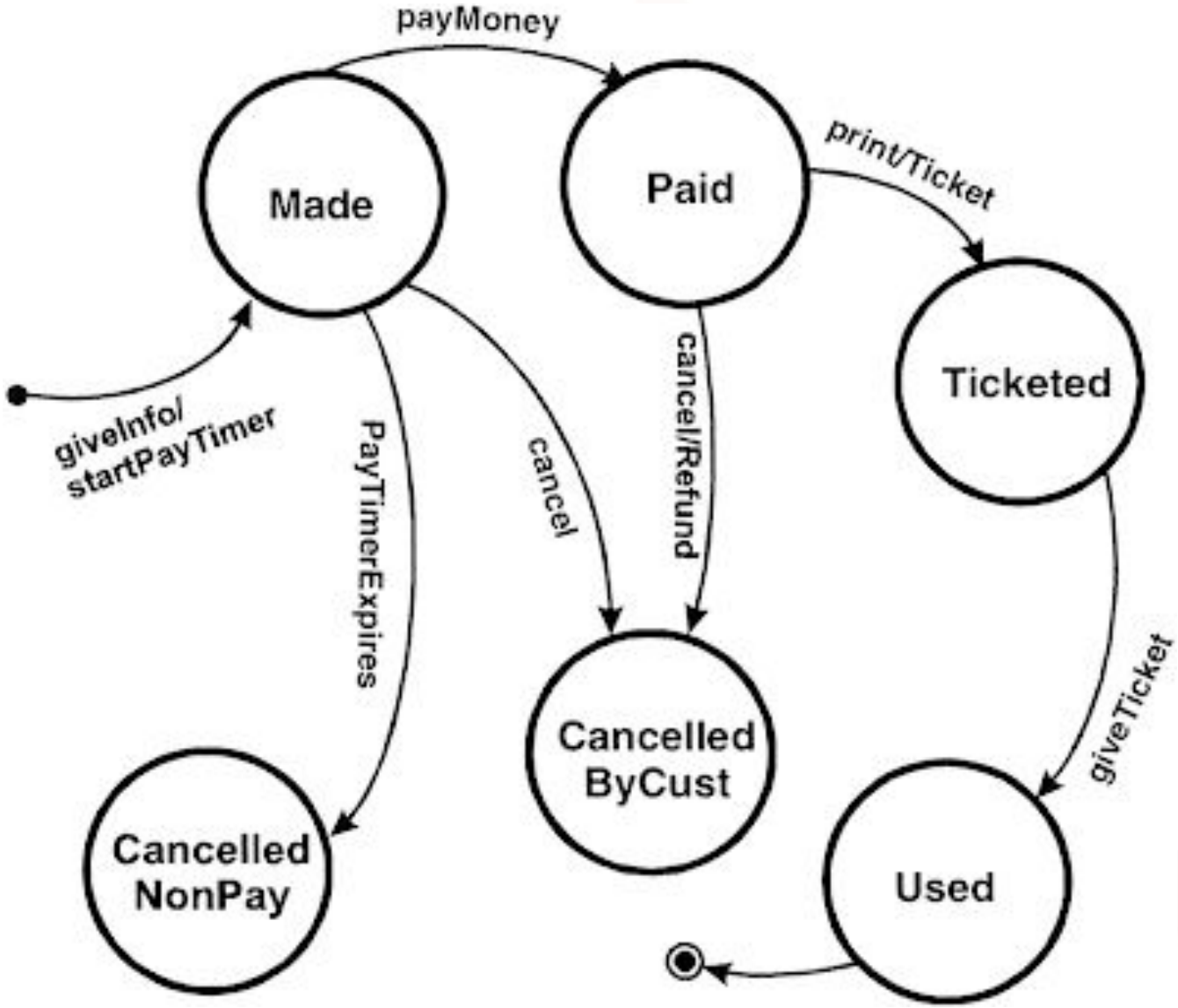
A-Level UKRAINE



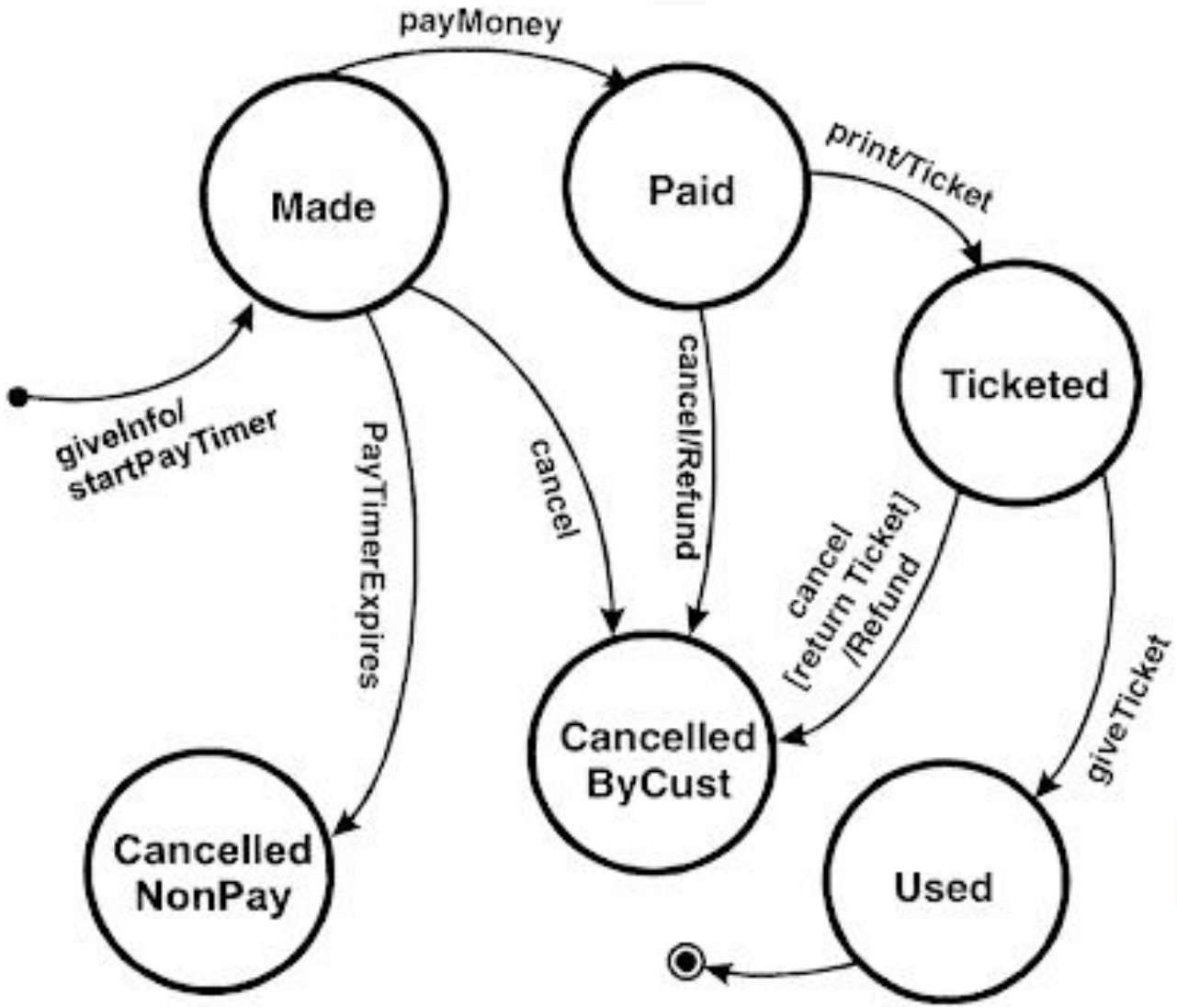
Якщо резервація не оплачена після закінчення таймера, вона скасовується як і оплачена.



Іноді клієнти скасовують замовлення зі стану "Made". На цю справу нам потрібен ще 1 стан "Cancelled By Customer".

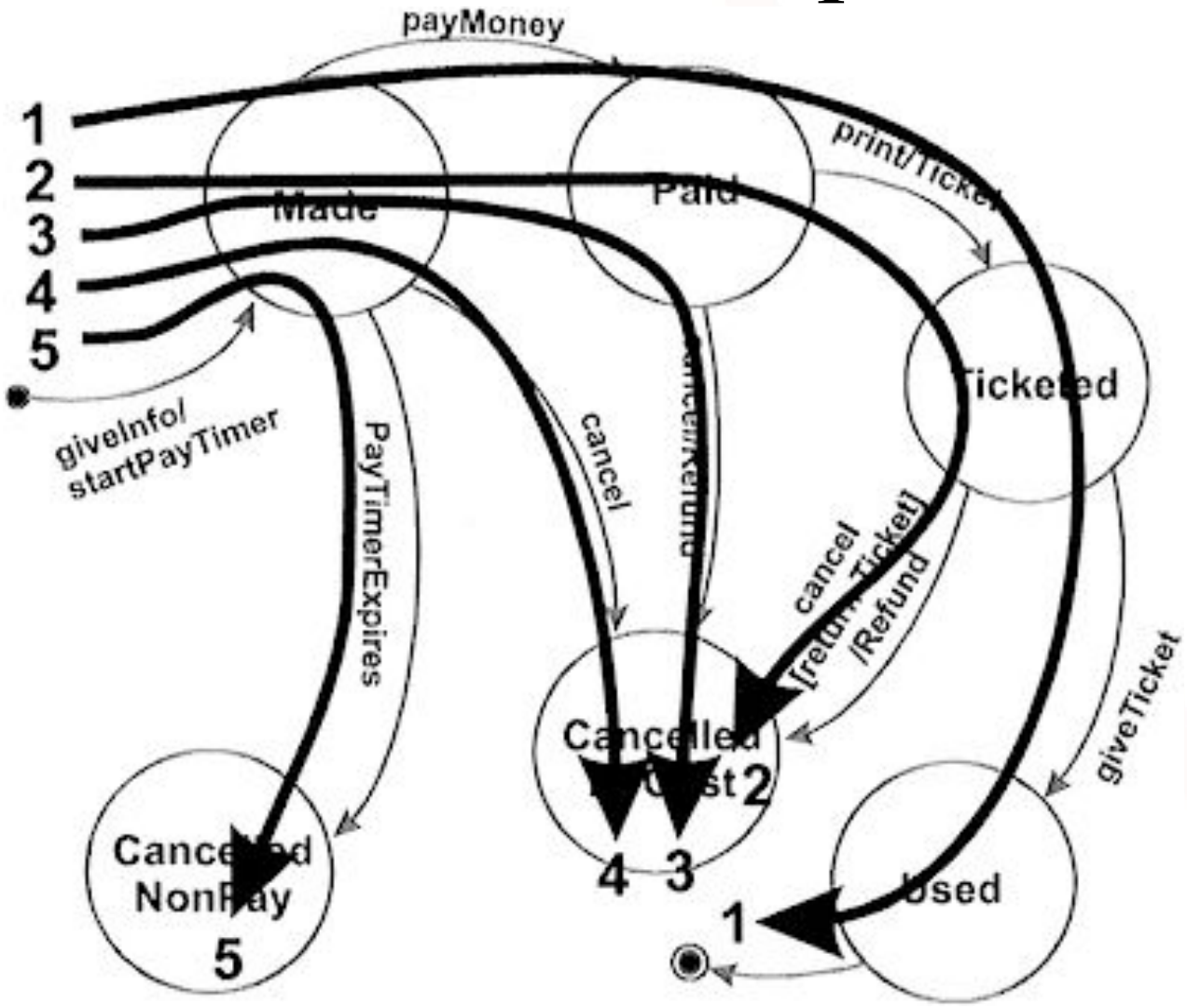


Іноді клієнти скасовують замовлення зі стану "Made". На цю справу нам потрібен ще 1 стан "Cancelled By Customer".



Скасувати резервацію можна і зі стану "Ticketed". І тут стан знову ставати " Cancelled By Customer " і авіакомпанія відшкодовує вартість квитка клієнту. Але! Компанія відшкодує вартість квитка лише тоді, коли клієнт поверне квиток. Цей випадок представляє ще один не описаний елемент діаграм - квадратні дужки "[]", які представляють умову переходу. Перехід у стан у цьому випадку трапиться тільки якщо умова (що в "[]") = true.

Створення тест-кейсів



State-Transition Diagrams можна легко використовувати для створення тест кейсів. Необхідно створити набір тест-кейсів, який має пройти по всіх переходах хоча б разів.

Попарне тестування

Pairwise testing (попарне тестування) – це техніка формування наборів тестових даних із повного набору вхідних даних у системі, яка дозволяє суттєво скоротити кількість тест-кейсів.

Сформулювати суть попарного тестування можна так: формування таких наборів даних, у яких кожне тестоване значення кожного з параметрів, що перевіряються, хоча б один раз поєднується з кожним тестованим значенням всіх інших параметрів, що перевіряються.

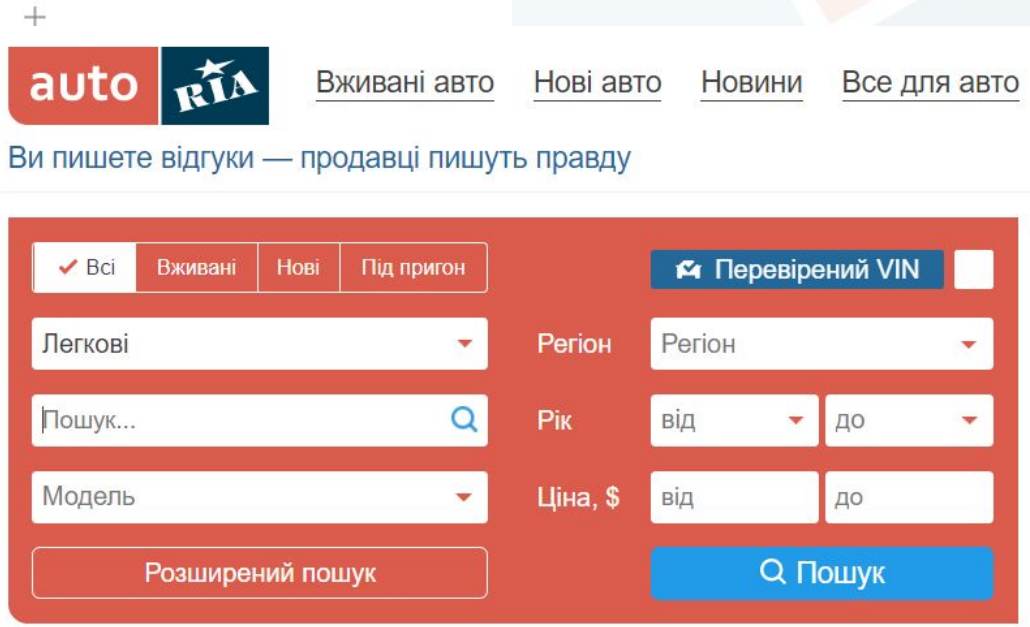
Головні цілі Pairwise Testing:

- усунути надлишкові перевірки;
- забезпечити гарне тестове покриття;
- виявити найбільшу кількість багів на мінімальному наборі тестів.

A-Level UKRAINE

pairwiseTool | Edit Conditions | Generate Pairwise | Generate All Combinations | Create Permalink

	Тип автомобиля	Марка	Регион
Row 1	Легковой	Audi	Киев
Row 2	Moto	BMW	Днепр
Row 3	Грузовик	Skoda	Харьков
Row 4	Автобус	Honda	Полтава
Row 5	{your value}	Kia	{your value}
Row 6	{your value}	Mazda	{your value}



A-Level UKRAINE

	Тип автомобиля	Марка	Регион
1	Легковой	Audi	Киев
2	Легковой	BMW	Днепр
3	Легковой	Skoda	Харьков
4	Легковой	Honda	Полтава
5	Мото	BMW	Харьков
6	Мото	Skoda	Полтава
7	Мото	Honda	Киев
8	Мото	Kia	Днепр
9	Мото	Mazda	Киев
10	Мото	Audi	Днепр
11	Грузовик	Skoda	Киев
12	Грузовик	Honda	Днепр
13	Грузовик	Kia	Киев
14	Грузовик	Mazda	Днепр
15	Грузовик	Audi	Харьков
16	Грузовик	BMW	Полтава
17	Автобус	Honda	Киев
18	Автобус	Kia	Днепр
19	Автобус	Mazda	Харьков
20	Автобус	Audi	Полтава
21	Автобус	BMW	Киев
22	Автобус	Skoda	Днепр
23	Легковой	Kia	Харьков
24	Легковой	Mazda	Полтава
25	Мото	BMW	Киев
26	Мото	Skoda	Днепр
27	Мото	Honda	Харьков
28	Мото	Kia	Полтава

Домашнє завдання

1. Конспект;
2. Перейти до сторінки (https://pairwise.teremokgames.com/);
3. Відкрити каталог будь-якої групи товарів в інтернет-магазині Citrus;
4. У pairwiseTool додати всі параметри з фільтра для обраної групи товарів;
5. Перенести значення для кожного параметра (крім ЦІНИ);
6. Створити Permalink;
7. Сформувати всі можливі набори значень тестування;
8. - Сформувати набори значень для тестування з використанням техніки попарного тестування;
9. Створене посилання та результати згенерованих комбінацій додати до word-документа;
10. Завантажити документ на платформу.

A-Level UKRAINE

Корисні посилання

1. https://training.qatestlab.com/blog/technical-articles/pairwise-testing/
2. https://training.qatestlab.com/blog/technical-articles/equivalence-classes-and-boundary-values/
3. "A Practitioner'S Guide To Software Test Design" by Lee Copeland.

A-Level UKRAINE