



Техники за дизайн на тест сценарии

Статични техники

Статични техники

Статичното тестване е техника, при която софтуера се тества без да се изпълнява кода, чрез така наречените ревюта или статични анализа на кода

- **Ревю:** обикновено се използва , за да се открият грешки в документи: изисквания, дизайн, тест сценарии и др.
- **Статични анализи:** кодът написан от разработчиците се анализира (обикновено с инструменти за статичен анализ) за да се открият структурни дефекти, които могат да доведат до грешки при изпълнението



Статични техники

Ревю

Цели: Верификация срещу спецификации и стандарти

Активности:

- Планиране (planning)
- Подготовка (preparation)
- Среца (review meeting)
- Последващи действия (follow-up)



Статични техники

Роли

1

Moderator (Модератор) неутрален участник, който подпомага гладкото преминаване на ревюто . Той играе още ролята на медиатор при възникнали спорове. Обикновено е човек, който има специална подготовка да изпълнява тази роля

2

Author (Автор): авторът на документа или кода, който е предмет на дискусията. По време на срещата той отговоря на въпроси свързани с документа, както изяснява неяснотите .

3

Readers/ Reviewers : по време на срещата тези участници подлагат на обсъждане документа или кода. Целта е да се открият предположения, липса на документация, лош стил и други грешки, които авторът не е способен да открие сам

4

Scribe: води записки по време на срещата като отбелязва всички потенциални дефекти. Целта е останалите участници да са освободени от това задължение, за да се съсредоточат върху обсъждането



Статични техники

Ревю



Статични техники

1

Informal review / Code review:

Това е тип ревю, което се прави от колега от екипа - най-често това е колега тестер или разработчик или лидера на екипа. Целта е да се провери дали документа има всички необходими елементи: анотация, въведение, описание на контекста и др. Ако това е програмен код дали има коментари, дали е форматиран правилно и тн.

Обикновено при такива тип ревюта се изчиства езика, на който е написан документа от жаргонни думи, чуждици и др и се определя дали авторът на документа предполага, че неговите читатели знаят повече, отколкото те в действителност знаят.

2

Walkthrough / Brainstorming:

Тази техника представлява вид среща, която се води от автора на документа. Целта е идеите или подходът на автора да се споделят с по-широка аудитория. Още се нарича brainstorming. Участниците трябва да са подготвени за срещата: те трябва да са запознати с целите на срещата и обикновено са чели документа преди срещата да се проведе. Срещата има точно определено време за провеждане и участниците често имат определени роли.

3

Technical Peer Review:

Освен членове на екипът при този вид ревюта задължително участва и технически лица, може да няма представители на менижмънта. В идеалния случай участва модератор. Основните цели са да се дискутира, да се вземат решения, да се обсъдят алтернативи, да се открият дефекти, да се разрешат технически проблеми, да се провери съответствие със стандарти и спецификация.

4

Inspections:

Този вид ревю е най-формален от всички. На него задължително има представители на менижмънта. Участниците трябва да са обучени преди да участват в такава среща. Тя се води от модератор и има точно определено време за провеждане обикновено 2 часа. Откритите дефекти се репортват – няма дискусия.



Статични техники

Ревю

Ползи

- Спестяват време и пари
- Откриват се и се отстраняват дефекти в много начална фаза
- Членовете на екипа са по-внимателни и имат по-добро разбиране какво се очаква от тях - елиминират се допусканията




Статични техники

Статични анализи

Това е статична техника, при която кодът се анализира, за да се открият грешки в синтаксиса и за да се генерира статистика за кода. При тази техника кодът не се изпълнява, освен това се предполага, че той изпълнява бизнес функционаностите, за които е предназначен, защото тази техника не може да открие такива несъответствия. Чрез тази техника се откриват синтактични грешки както и нарушения на някои стандарти или добри практики. Някои инструменти за статичен анализ могат да дадат оценка на сложността на програмата, както и да дадат предложения как тя да се намали.

Съществуват множество инструменти за статичен анализ

Грешките които могат да се открият с такива инструменти са:

- 
- Използване на неинициализирани променливи
 - Несъвместими интерфейси
 - Променливи, които никога не се използват
 - Недостъпен код (dead code)
 - Нарушение на стандарти
 - Синтактични грешки

Статични техники

Статични анализи

Компилатори :

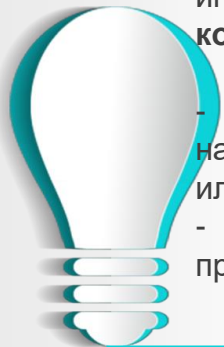
Съществуват 2 вида езици за програмиране: интерпретативни и не-интерпретативни (изискващи компилация) . Компилаторите се използват , за да превърнат кодът написан на съответния език в двоичен код. Преди да компилират кодът, компилаторите правят синтактична проверка на кода ,за да се отстранят всички синтактични грешки – например липсващи скоби. В допълнение те могат да допълнителна информация за кода , променливите и използваната памет.

Статични анализатори:

Това са програми, които са се появили като еволюция на компилаторите ,като дават по детайлна информация и правят по- детайлни метрики за него. **Те за разлика от компилаторите обаче не компилират кода.**

- **Data flow analyzes:** това е анализ на потока на данни в програма, чрез проследяване на данните, може да се открие дали има променливи, които се използват преди да са инициализирани или да се достъпят такива, които вече не съществуват

- **Control flow analyzes:** това е анализ на начина, по който контролът се предава през програмата. Използва се, за да се открият безкрайни цикли, недостъпен код.



Статични техники

Статични анализи

Метрики:

Всички автоматизирани инструменти за статичен анализ предлагат различни статистики за кода – например:

- **Брой редове** : Използва се за да се определи размера на метод, клас, програма или цялото приложение.
- **Nesting level**(брой IF, които са вметени в други IF)
- **Cyclomating complexity** (броят на независимите пътища в метод или програма) . Обикновено се използва на ниво метод . Стойности под 10 се смятат за приемливи за един метод, между 10 и 20 трябва да са рядкост, а над 20 да се избягват напълно.
- **-Брой методи/променливи в клас**. Стойности от около 20 се смятат за нормални
- **-Брой параметри/променливи в метод**.



Динамични техники

Black box testing

Тази техника се базира на функционалността на компонента. С други думи при нея се интересува какво прави този компонент, а не какво го прави.

Пример1: Ел. Крушка: ако използваме ключа, за да я запалим , ще можем да проверим дали тя свети, без да имаме задълбочени знания за електричеството и законите на Кирхов.

Пример2: В интерфейса на дадено приложение има поле за въвеждане на текст:

То приема стойности от +1 до +100

- Има 100 валидни теста, които трябва да се направят , използвайки числата от 1 до 100
- Има безкраен брой невалидни тестове
- Има безкраен брой от невалидни тестове, използвайки букви или комбинации от букви и цифри

Очевидно е ,че не могат да се извършат всички възможни тестове с всички възможни комбинации, за това се използват няколко познати техники , които да ни позволят да създадем разумен и управляем набор от тестове , който да покрива всички сфери, като се фокусира върхи сферите, в които могат да възникнат грешки.



Динамични техники

Black box testing

Тази техника се базира на функционалността на компонента. С други думи при нея се интересува какво прави този компонент, а не какво го прави.

Пример1: Ел. Крушка: ако използваме ключа, за да я запалим , ще можем да проверим дали тя свети, без да имаме задълбочени знания за електричеството и законите на Кирхов.

Пример2: В интерфейса на дадено приложение има поле за въвеждане на текст:

То приема стойности от +1 до +100

- Има 100 валидни теста, които трябва да се направят , използвайки числата от 1 до 100
- Има безкраен брой невалидни тестове
- Има безкраен брой от невалидни тестове, използвайки букви или комбинации от букви и цифри

Очевидно е ,че не могат да се извършат всички възможни тестове с всички възможни комбинации, за това се използват няколко познати техники , които да ни позволят да създадем разумен и управляем набор от тестове , който да покрива всички сфери, като се фокусира върхи сферите , в които могат да възникнат грешки.



Динамични техники

Black box testing

Equivalence partitioning:

Тази техника се базира на предположението, че входните и изходните данни на компонента могат да се разделят на класове, които по спецификация ще бъдат третирани по един и същи начин от приложението.

При тази техника първо се идентифицират класовете, след което се подготвят тестове, които да тестват всеки от идентифицираните класове.

Пример: Бизнес правило: Кредит се дава за период между 6 и 10 години.

Разделението на класове е следното

0	1	2	3	4	5		6	7	8	9	10		11	12	13	14	
Invalid							Valid						Invalid				

Идентифицирани са 2 класа с невалидни стойности: 0-5 и 11-14 и един клас с валидни стойности 6-10.

Използвайки тази техника, би трябвало да избелерем по една стойност от всеки клас и да тестваме с нея . Например 4, 7 и 12



Динамични техники

Black box testing

Boundary value analyzes.

При тази техника отново се прави разделение на класове, но се тестват граничните стойности между класовете – включително минимални и максимални.

Пример: Полето парола може да бъде между 8 и 12 символа

**Invalid
Partition**

Less than 8

**Valid
Partition**

8 - 12

**Invalid
Partition**

More than 12



Възможните тестове ще са :

- Парола по-къса от 8 символа (например 7)
- Парола точно 8 символа
- Парола между 9-11 символа (например 10)
- Парола точно 12 символа
- Парола по-дълга от 12 символа (например 17)

Динамични техники

Black box testing

Задача1: За да си подходящ за отпускане на ипотечен кредит, трябва да си на възраст между 18 и 64 години включително. Полето приема само двуцифрени числа и не приема знак минус. Кой за валидни и невалидни стойности за Boundary analyzes и Equivalence partitioning.



Динамични техники

Black box testing

Задача2: Потребителски кредит се отпуска за суми между 5 000 и 70 000 лв. Кредити между 5 000 – 25 000 лева са с лихва 8,5% , а кредити между 25 и 70 хиляди са с лихва 8% . предложете тест сценарии използвайки Boundary analyzes и Equivalence partitioning.



Динамични техники

Black box testing

Decision table testing

Пример:

Банкомат: клиентът задава сума. Банкоматът трябва да изплати сумата, ако клиентът има достатъчно пари по сметката си или ако разрешеният му овърдрафт не е достигнат.



Динамични техники

Black box testing

Decision table testing

Стъпка 1: Анализ на изискванията и създаване на първата колона

Условие
Сума <= Баланса
Кредит
Действие
Сумата е позволена

В случая имаме две условия

Исканата сума <= Баланса

Има овърдрафт

Действието е само едно – **Отпускане на сумата**

Условие	R1	R2	R3
Сума <= Баланса	T	F	F
Кредит	-	T	F
Действие			
Сумата е позволена	T	T	F



Динамични техники

Black box testing

Decision table testing

Стъпка 2: Добавяне на колони: Броят на колоните се определя по формулата $2^{\text{condition}}$.в нашия пример $2^2 = 4$

Условие				
Сума<=Баланса	T	F	T	F
Кредит	T	T	F	F
Действие				
Сумата е позволена				



Динамични техники

Black box testing

Decision table testing

Стъпка 3: Редуциране на колоните

Когато първото условие е достатъчно, за да се вземе решение, второто няма нужда да бъде проверяването

Условие				
Сума <= Баланса	T	F	T	F
Кредит	-	T	-	F
Действие				
Сумата е позволена				



Динамични техники

Black box testing

Decision table testing

Стъпка 4: Определяне на действията или очакваните резултати

Условие			
Сума <= Баланса	T	F	F
Кредит	-	T	F
Действие			
Сумата е позволена	T	T	F



Динамични техники

Black box testing

Decision table testing

Стъпка 5: Разписване на test cases

Условие	R1	R2	R3
Сума<=Баланса	T	F	F
Кредит	-	T	F
Действие			
Сумата е позволена	T	T	F

R1: Баланс=200 , Искана сума=200 . Очакван резултат : сумата е отпусната

R2: Баланс=100, Искана сума = 200, Овърдрафт е разрешен. Очакван резултат: сумата е отпусната

R3: Баланс=100, Искана сума= 200. Няма разрешен овърдрафт. Очакван резултат: сумата не е отпусната



Динамични техники

Black box testing

State transition

Това е техника за black-box тестване, който се използва, когато имаме така наречената Машина на състоянията – state machine. Системата може да се намира в краен брой състояния и преминаването от едно състояние в друго се определя от „правилата“ на машината.

Дефинират се 4 основни части на Машината на състоянията:

- Състояния (States)
- Преходи (transitions)
- Събития, които предизвикват прехода от едно състояние в друго
- Действия, които произтичат от прехода

Едно събитие може да предизвика само едно действие, но същото действие от различно състояние , може да предизвика друго действие и преминаване в друго състояние.

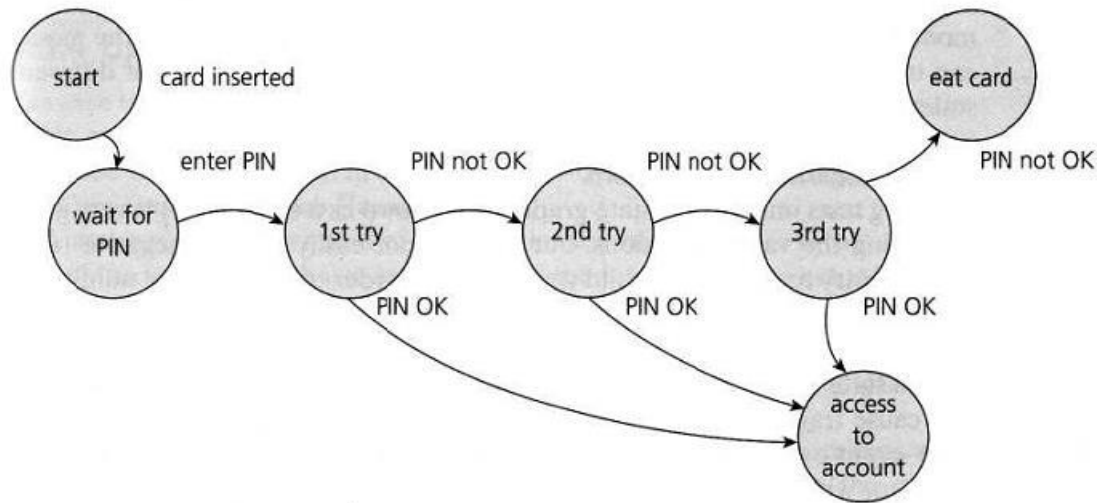
Ако се върнем на примера с банкомата: ако поискаме 100 лв, то парите могат да ни бъдат отпуснати, но системата може да влезе в друго състояние – недостатъчен баланс и при следващ опит за теглене, сумата да не може да бъде изтеглена вече и да се получи отказ.



Динамични техники

Black box testing

State transition



Динамични техники

Black box testing

State transition

Тест1: Коректен ПИН е въведен от първия път

Тест2: Некоректен ПИН се въвжда 3 пъти, така че системата да задържи картата

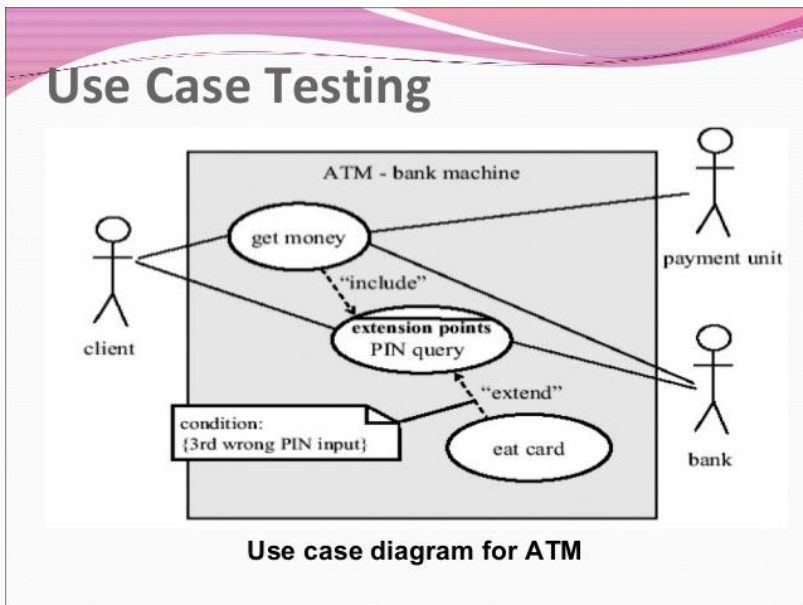
Тест3: Некоректен ПИН първи път , коректен ПИН втори път, така че достъп до сметката е разрешен



Динамични техники

Black box testing

Use cases



Use case: описание на определен начин за използване на системата от потребител в определена роля. Всеки use case описва взаимодействие на протребителя със системата, с цел да изпълни някаква задача. Потребителите обикновено са хора, но могат да бъдат и други системи.

Use cases са набор от стъпки, които опиват взаимодействията на потребителя със системата, но не като входни и изходни данни, а в от гледна точка на това какво потребителът прави и какво вижда. Обикновено използваните термини са от Бизнес света, а не толкова технически. Все use case има най-често използван сценарии и алтернативни разклонения, покриващи специфични случаи.



Динамични техники

Black box testing

Use cases

Main Success Scenario A: Actor S: System	Step	Description
	1	A: Inserts card
	2	S: Validates card and asks for PIN
	3	A: Enters PIN
	4	S: Validates PIN
	5	S: Allows access to account
Extensions	2a	Card not valid S: Display message and reject card
	4a	PIN not valid S: Display message and ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat card and exit



Динамични техники

White box testing

Това са техники, които се базират на познание на вътрешната структура на компонента или системата. В различните нива на тестване различни структурни елементи са важни:

- Component level: структурата е самия код
- Integration level: структурата е начина, по който отделните модули са комбинирани
- System level: структурата са бизнес процеси

При тези техники не се интересуваме от функционалността на приложението, а само от броя на разклоненията, които са покрити с тестове – **Test coverage**

Тези тестове са **автоматизирани**, защото не могат да бъдат изпълнявани самостоятелно.



Динамични техники

White box testing

Това са техники, които се базират на познание на вътрешната структура на компонента или системата. В разните нива на тестване различни структурни елементи са важни:

- Component level: структурата е самия код
- Integration level: структурата е начина, по който отделните модули са комбинирани
- System level: структурата са бизнес процеси

При тези техники не се интересуваме от функционалността на приложението, а само от броя на разклоненията, които са покрити с тестове – **Test coverage**

Тези тестове са **автоматизирани**, защото не могат да бъдат изпълнявани самостоятелно.



Динамични техники

White box testing

Statement coverage:

Това е white-box техника, при която всеки statement в кода трябва да се изпълни поне веднъж.

$$\text{Statement coverage} = \frac{\text{Number of statements exercised}}{\text{Total number of statements}} \times 100\%$$



Динамични техники

White box testing

Statement coverage:

Пример:

- 1.Read A
- 2.Read B
3. if A>B
4. Print "A is greater than B"
- 5.else
- 6.Print "B is greater than A"
7. endif

Set1 :If A =5, B =2

No of statements Executed: 5

Total no of statements in the source code: 7

Statement coverage $= 5/7 * 100 = 71.00 \%$

Set2 :If A =2, B =5

No of statements Executed: 6

Total no of statements in the source code: 7

Statement coverage $= 6/7 * 100 = 85.20 \%$



Динамични техники

White box testing

Branch/Decision coverage

Branch е изходът от дадено решение, branch coverage-а мери кои разклонения на алгоритъма са покрити.

$$\text{Decision coverage} = \frac{\text{Number of decision outcomes exercised}}{\text{Total number of decision outcomes}} \times 100\%$$



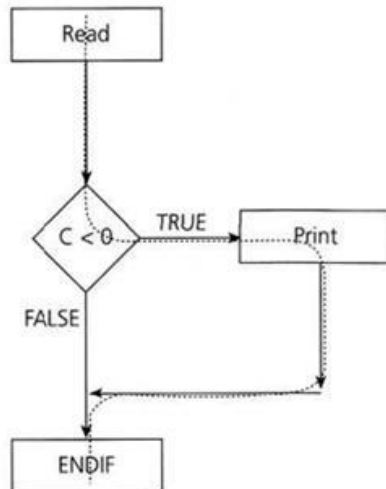
Динамични техники

White box testing

Statement coverage:

Пример:

```
1 READ A
2 READ B
3 C = A - 2 * B
4 IF C < 0 THEN
5 PRINT "C negative"
6 ENDIF
```



Set 1: A = 20, B = 15

100% Statement coverage
50% Branch coverage

Set 1: A = 20, B = 15

Set 2: A = 10, B = 2

100% Statement coverage
100% Branch coverage

