**Лекция 4** Тестови техники

Ползваме тестови техники за да:

Бъдем по - ефективни в създаването на тест кейси.

За да сме сигурни, че покриваме софтуера по правилния начин.

Ръчния тестинг е най – популярния метод за валидация на софтуер.

**Black box** техники - познато като тестване базирано на спесификации, интересуват ни само входните и изходните данни, не и как се изчисляват от софтуера.

Equivalence Partitioning: EP

Входните данни се разделят на групи (партишъни), които се очаква да покажат еднакво поведение.

Партищъните могат да бъдат валидни и невалидни. Една от най-ползваните техники, намалява повтаремоста и бройката на тестовете.

Партишъните НЕ ТРЯБВА да се припокриват/застъпват и не трябва да са празни.

**---------------**

Baundary value analysis: BVA

Фокусира се върху определяне на границите на входните стойности и тества върху тези избрани граници.

Най-ползваните техники, софтуера обикновенни се чупи при граничните стойноти. Много ефективен за намиране на бъгове. Комбинира се със EP.

Един Equivalence class (партишън) може да има една или две граници или наречени още долна и горна.

Прилагане на BVA:

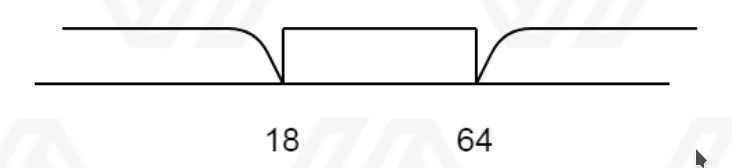
1. Долна граница (ако съществува)

2. Горна граница (ако съществува)

3. --Всяка стойност вътре в дадения партишън/клас (ако в него има 3 или повече стойности).-- това ре за партишъните

4. Горната граница на класа преди този който тестваме (ако има преден клас/партишън).

5. Долната граница на следващия клас след този който тестваме (ако има такъв).



В случая това ще са 17, 18,42(3. всяка стойост),64 ,65

**---------------**

Decision table testing: DT

Тества поведението за различни входни комбинации.

Различни входни комбинации и техните съответни изходи данни (output)

са представени в таблична форма.

DT е таблична репрезентация на входни данни

срещу правила/кейси/тест състояния (conditions).

State transition:

Changes in input conditions couse state change in software under test (SUT)’

Analyzing the behavior of SUT for different input conditions in a sequence.

Couse-Effect graph поллзваме:

За да индетифицираме причината на даден проблем.

За да анализираме съществуващи проблеми и предприемем действия за отстраняването им.

Error Guessing:

Отгатване на грешките въз основа на предишен опит.

**---------------------------------------------------------------------------**

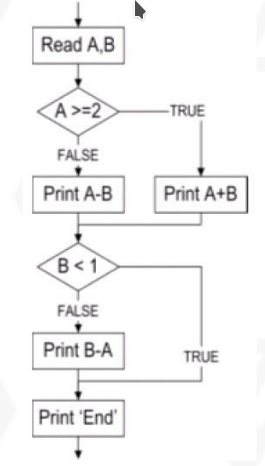
**White box** техника:

Whats under da hood man. :)

Тест при който вътрешната структура/код/изпълнение на дадения тестван обект са познати на тестера.

Statement coverage:

Изпълнение на всички изпълними стейтмъни (правоъгълници) в сорс кода.



Decision (Branch) coverege:

Изпълнение на всички бранчове (ромбове).

Всички бранч пътища трябва да са покрити.

Path coverege:

Всички възможни пътища в кода на тествания обект трябва да са покрити.

100% path coverege означава 100% statement and branch coverege.

Data flow testing:

Изпълнява се автоматично от програмата в която се пише кода.

Static data flow:

Програмата идентифицира потенциални дефект в кода докат бъде писан,

така че автора на кода да види и поправи дефекта.

Кода не се изпълнява.

Dynamic data flow:

Кодът се изпълнява и тогава засича и показва грешките допуснати при писането.

Последния IF се затваря с първия END IF