YAZILIM TESTİ VE PROJE YÖNETİMİ

Yazılım Testlerinin Sınıflandırmaları

2024 BAHAR

06.05.2024

Black-Box Functional Testing Techniques

- ❑Equivalence Partitioning
- ❑Boundary Value Analysis
- ❑Decision Table Testing
- ❑State Transition Testing
- ❑Use Case Testing

Karar Tablosu Testi (Kara Kutu Testi)

- Farklı giriş kombinasyonları için sistem davranışı test edilir.
- Farklı girdi kombinasyonlarının (input) ve bunlara karşılık gelen sistem davranışının (output) tablo olarak verildiği bir yaklaşımdır.
- Daha iyi test içeriği oluşturmak üzere sebep ve sonuçların belirlendiği Sebep-Sonuç (Cause-Effect) tablosu olarak da adlandırılır.
- Karar tablosu sadece yazılım testinde değil, yazılım gereksinimlerinin belirlenmesinde de kullanılır.

Karar Tablosu Test Örneği

Koşullar	Kural1/ TestCase1	Kural 2/ Test Case 2	Kural 3/ Test Case 3	Kural 4 / Test Case 4
email Id (Input)	T	T	F	F
şifre (Input)	T	F	T	F
eylem (Output)	H	E	E	E

Sistemin Çıktısı bir eylem bildirir. H ise kullanıcı başarılı giriş yapar, E ise kullanıcı girişi hatalıdır.

eposta giriş ekranının Karar Tablosu ile test edilmesi

Dosya Yükleme Ekranının Karar Tablosu ile Testi

Testin gerçekleştirilmesi için gerekenler girişlerdir (inputs)

Güncelleme ekranı için test edilmesi gerekenler:

i)Dosya .png formatındadır.

ii)Dosyanın uzunluğu 25 kb. den küçüktür.

iii)Dosyanın çözünürlüğü 132*170 pixel olmalıdır.

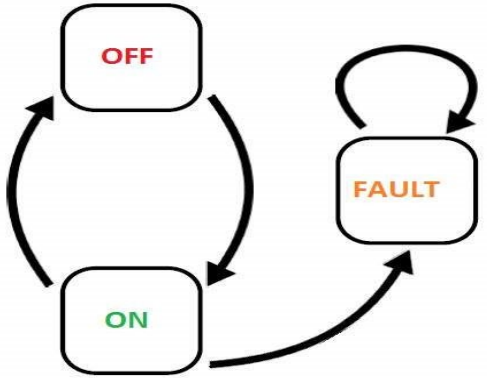
Sistemin çıktısı (output) ise 8 farklı test case ile irdelenecektir.

Çünkü 3 farklı veri girişi olduğu için 8 farklı test case (TC) tanımlanır (2^3).

Ekran Yüklemesinin (Screen Upload) Testinin Karar Tablosu ile Yapılması

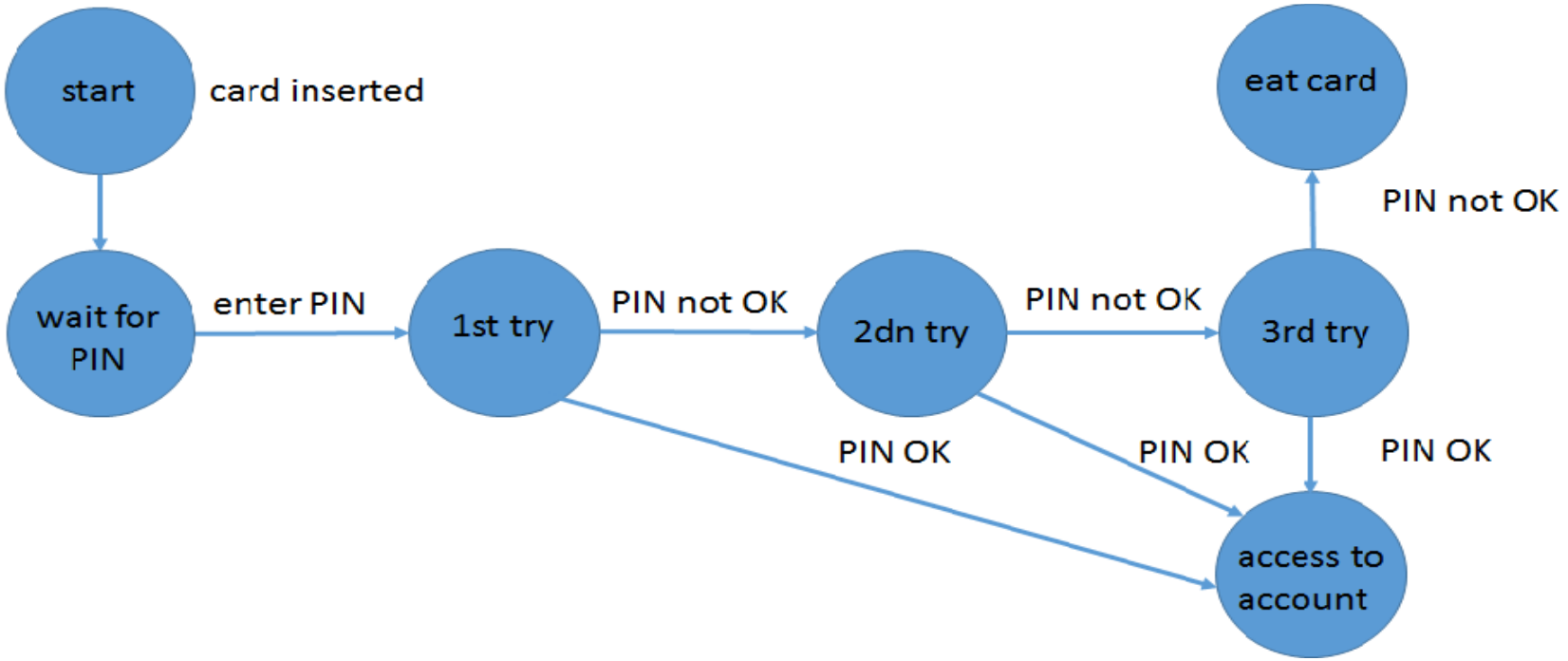
Koşullar	Kural 1/TC1	Kural 2/TC2	Kural 3/TC3	Kural 4/TC4	Kural 5/TC5	Kural 6/TC6	Kural 7/TC7	Kural 8/TC8
Format (input)	.png	.png	.png	.png	Not .png	Not.png	Not.png	Not.png
Büyüklük (input)	<25kb	<25kb	>=25kb	>=25kb	<25kb	<25kb	>=25kb	>=25kb
Çözünürlük (input)	= 132*170px	!= 132*170px	=132*170px	!= 132*170px	=132*170px	!= 132*170px	=132*170px	!= 132*170px
Eylem(Output)	.png dosyasının başarı ile yüklenmesi	Error mesajı çözünürlüğün uyumlu olmadığı mesajının yazması	Görüntü büyüklüğünün uyumlu olmadığı Error mesajının yazması	Görüntü büyüklüğünün uyumlu olmadığı Error mesajının yazması	Formatın uyumlu olmadığı Error mesajının yazması	Formatın uyumlu olmadığı Error mesajının yazması	Formatın uyumlu olmadığı Error mesajının yazması	Formatın uyumlu olmadığı Error mesajının yazması

Durum Geçiş Testi /State Transition Test
Kara Kutu Testi



- Sistemin farklı durumları arasında geçiş yaparken yazılımın beklendiği gibi davranmasını kontrol etmede kullanılır.
- 1. adımda, yazılımın sahip olabileceği sonlu durum kümesi tanımlanır.
- 1. adımda, yazılımın farklı giriş koşulları için, bunlar arasında nasıl geçiş yaptığı test edilir.
- Sistemin davranışları **sonlu durum makinesi** olarak tanımlanabilir.
  - Otomatlar, trafik ışıkları, web uygulamaları gibi faklı sistemlere uygulanabilir.
- Yazılımın karmaşık durum geçişleri incelenerek, yazılımın güvenilirliğini (**reliability**) ve sağlamlığını (**robusness**) sağlamak hedeflenir.
- Bu test, sistem bir **durumdan diğerine** geçerken oluşabilecek hataları tespit etmek için de kullanılabilir.
- Gerekli test senaryolarının sayısını azaltmada, test süresini optimize etmede ve yazılım kalitesini iyileştirmede iyi sonuçlar verir.

Durum Geçiş Testi / State Transition Testing
Black Box Functional Testing



- State Transition testing is a process-oriented test design technique
- State Transition testing focuses on states, events that initiate a transition to another state and actions resulting from such events.
- Tests are designed to execute valid and invalid state transitions.

Kullanım Senaryosu Testi
Use Case Testing

Success scenario	Step	Description
A – Actor S – System	1	A: Inputs login data
	2	S: Verifies login data
	3	S: Permits account access

- Use case (kullanım senaryosu) testi, yazılıma ait bir süreç (bir dizi işlem) bağlamında baştan sona tüm sistemi kapsayan test senaryolarının çıkartılmasıdır.
- Test senaryoları (test cases), kullanıcılarla uygulama arasındaki etkileşimlerdir.
- Bir uygulamaya ait yazılım bileşenlerinin tek tek test edilmesiyle belirlenemeyen durumların belirlenmesini sağlar.
- Teste ait bir kullanım durumu, uygulamanın aktör/kullanıcı tarafından belirli bir şekilde çalıştırılmasının kısa bir açıklamasıdır.
- Kullanım senaryoları, kullanıcı eylemlerine ve uygulamanın bu kullanıcı eylemlerine verdiği yanıtlara göre yapılır.
- Sistem veya kabul düzeyinde test senaryolarının geliştirilmesinde yaygın olarak kullanılır.

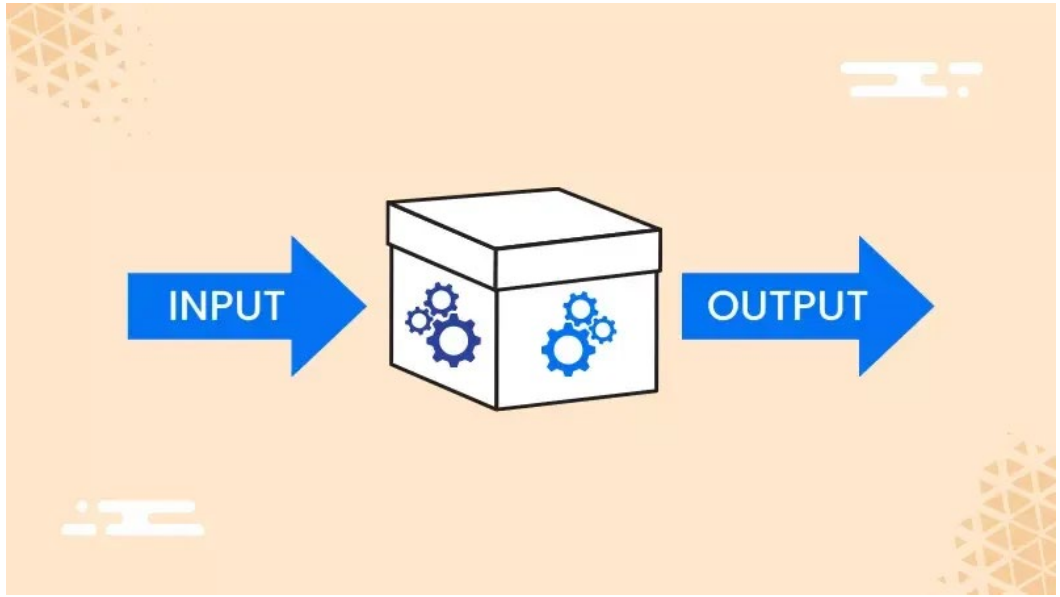
Basic flow	Step	Description
A – user S – application	1	A: inputs their login data
	2	S: verifies login data
	3	S: redirects the user to their account
	4	A: selects pictures
	5	A: presses ‘delete’ button
	6	S: displays action verification message
	7	A: confirms deletion
	8	S: removes selected pictures from the storage
	9	S: displays ‘deletion successful’ message
Extensions	1a	A: inputs incorrect login data
	2a	S: displays error message
Exception	7a	A: does not press ‘confirm’ button S: returns the user to their storage main page

Software Testing Knowledge Base

https://www.qamadness.com/knowledge-base/a-brief-guide-to-use-cases-and-use-case-testing/

Beyaz Kutu Testi- Yapısal /Structural Test- Kod Tabanlı Test

- Yazılım testinde kodun iç yapısını ve tasarımını kontrol eden test sürecidir.
- Test uzmanının kaynak koda erişimi vardır.
  - Yazılımın mantığını, akışını ve yapısını test eder.
- Test uzmanı, belirtilen gereksinimlerin karşılandığını kontrol amacıyla kod yollarını ve mantık akışlarını inceler ve bunun için test senaryoları oluşturur.
- **Temel amaç;** yazılımın kullanılabilirliğini, tasarımını ve güvenliğini iyileştirmektir.
- Özetle yazılımın genel performansını artırmak için veri yapısı, çalışma ayrıntıları ve program tasarımı test edilir.



  - Test uzmanları bu test için test senaryoları oluşturduklarından yazılım kaynağını ve mimarisini bilmelidirler.

Beyaz Kutu Test Teknikleri

- Statement Coverage Testing
- Decision Coverage Testing
- Branch Condition Testing
- …….

Statement Coverage Testing

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

```
print (int a, int b) {
int sum = a+b;
if (sum>0)
print ("This is a positive result")
else
print ("This is negative result")
}
```

Test Case 1:

a = 5, b = 4

Total number of statements = 7

Number of executed statements = 5

Statement coverage = 5/7*100 = 500/7 = 71%

Test Case 2:

a = -2, b = -7

Total number of statements = 7

Number of executed statements = 6

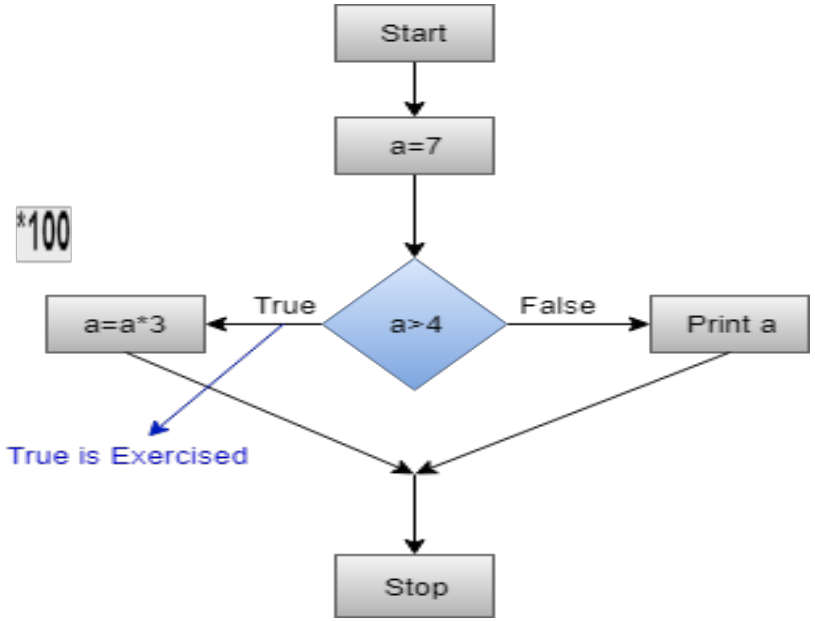
Statement coverage = 6/7*100= 600/7 = 85%

Decision Coverage Testing

- do while statement, if statement ve case statement bu test için verilmiş olmalıdır.
- Özetle «control flow statements» mevcut olmalıdır.

```
Test (int a)
{
If(a>4)
a=a*3
Print (a)
}
```

$$\text{Decision Coverage} = \frac{\text{Number of Decision Outcomes Exercised}}{\text{Total Number of Decision Outcomes}} * 100$$



Test Case 1: a=6
Decision Coverage = ½*100
(Sadece "True" çalıştı) =100/2 = 50
Decision Coverage is 50%

Test Case 2: a=2
Decision Coverage = ½*100
(Sadece "False "çalıştı) =100/2 = 50
Decision Coverage = 50%

Test Case	A değeri	Çıktı	Decision Coverage
1	2	2	50%
2	6	18	50%

Decision Coverage Testing
genel açıklamalar

- Bu test grubu, daha fazla «test cases» gerektirir. Ama:
- Beyaz kutu testlerinde «statement coverage test» yaklaşımından daha gelişmiş bir test kriteridir.
- Kaynak kod ile ilgili olarak alınacak kararlar test sürecinin odağındadır.
  - Karar alındığında etkisi belirlenir. Bu da, bu kararın bir sonraki adımda hangi ifadeleri çalıştıracağıdır.
  - Bu durum test süresince devam eder.
- Koda ait olarak alınan karar IF-CASE deyimleri, döngüler ve bunlara benzer ifadeler içindir.
  - Karar «true» veya «false» sonucun döndürüleceğidir.

Branch Condition Testing

- Branch Condition testinde daha düşük düzeydeki alt koşullar tanımlanır.
- Bu da hem «true», hem de «false» değerler için koşulların birlikte test edilmesidir.
- Bir koşul AND, OR ya da NOT gibi mantıksal (logical) koşullar içermez.
  - Ayrıntılar içerebilir. Bu nedenle de “>” ya da “=” gibi ilişkisel (relational) operatörleri içerebilir.
- Test nesnelerindeki (test cases) tek bir karar ya da çoklu koşullardan oluşabilir.

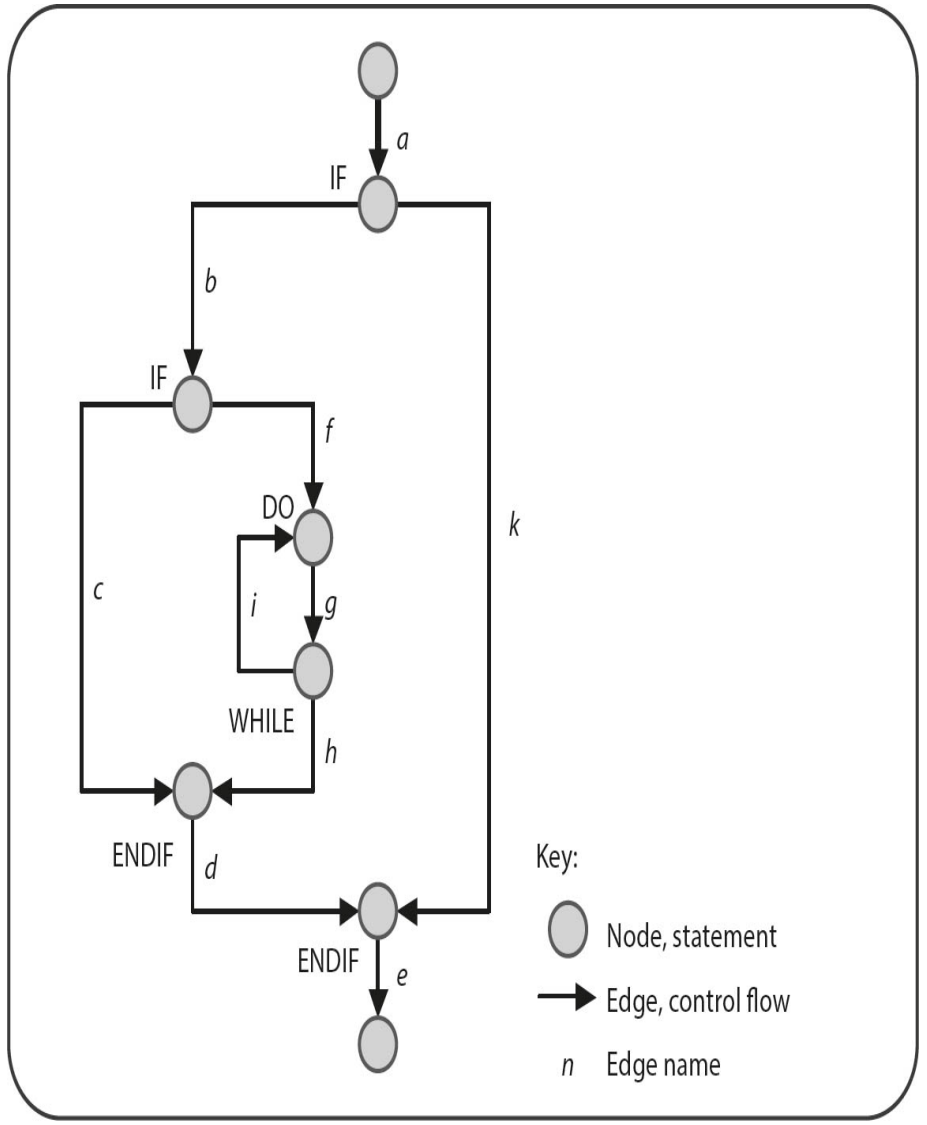
Statement Coverage ve Branch Coverage Testing

- **100% statement coverage :** a, b, f, g, h, d, e yolu olarak verilir. Bu sonuç kabul edildiğinde:
- c, i, k edges(kenarları) «statement coverage» test case içinde çalıştırılmayacaktır. Burada:
  - c ve k kenarları problemin false durumları (If deyiminin) , i edge ise döngünün başa dönmesidir.
- Bu 2 durum (if deyiminin false kısmı ve while döngüsü) , Statement Coverage değerlendirmesi içerisinde alınmamıştır.

100% branch coverage sağlanması için üç «test case» gereklidir. Her koşulun hem T hem de F test edilmesi gerekir.

Test case 1: a, b, c, d, e Test case 2: a, b, f, g, i, g, h, d, e

Test case 3: a, k, e (döngü 1 kere test edildi)



Kontrol Akış Diyagramı

Branch Coverage Testing

```
Test  (int a) {
    If (a> 5)
          a=a*3
    Print (a)
   }
```

$$Branch\ Coverage = \frac{Number\ of\ Executed\ Branches}{Total\ Number\ of\ Branches}$$

Test Case	Value of A	Output	Decision Coverage	Branch Coverage
1	2	2	50%	**33.3%**
2	6	18	50%	**66.6%**