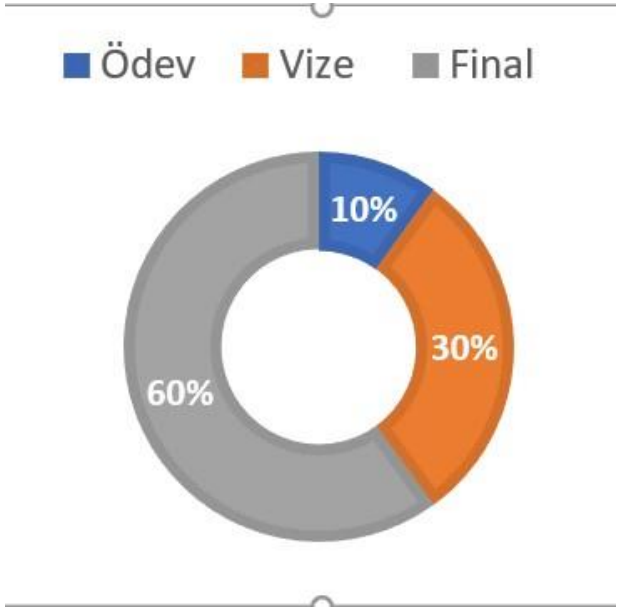
Yazılım Mühendisliği
1906003082015

Dr. Öğr. Üy. Önder EYECİOĞLU
Bilgisayar Mühendisliği

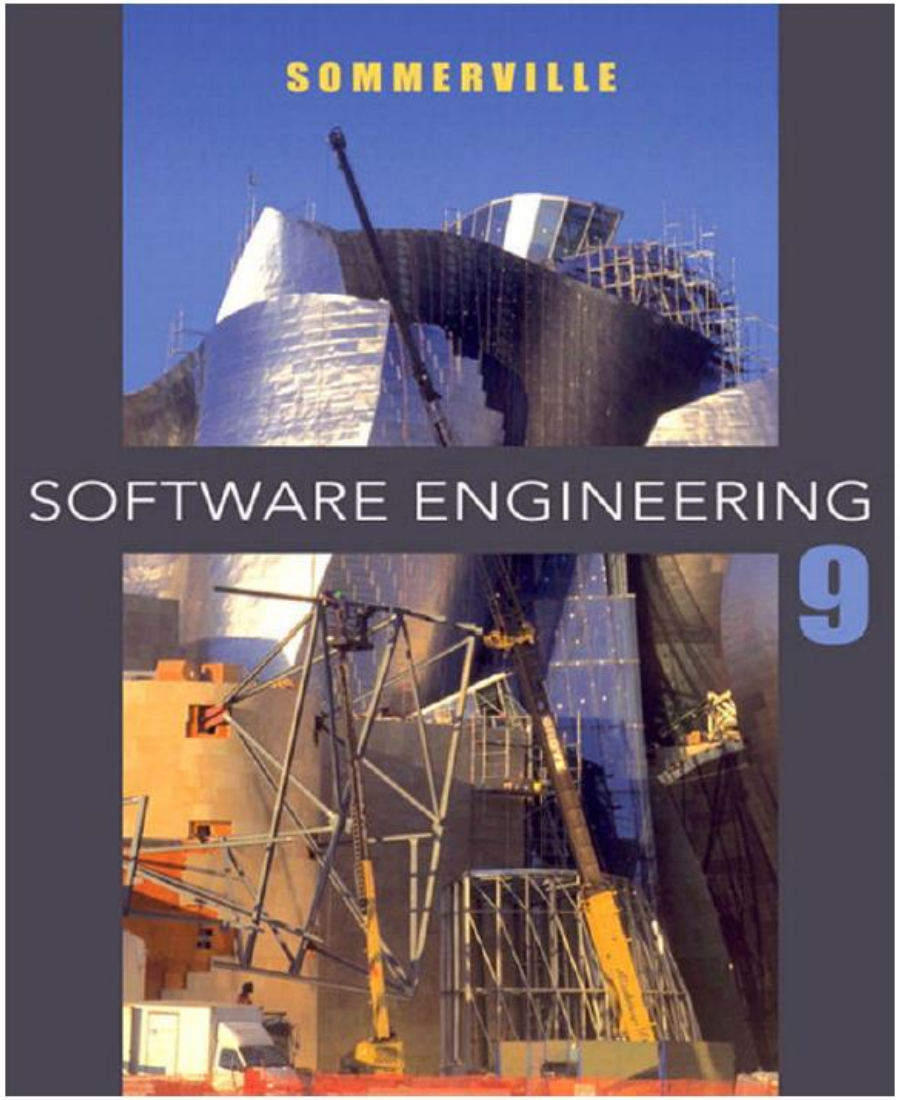
Giriş

Ders Günü ve Saati:
Salı: 09:15-13:00
Devam zorunluluğu %70



HAFTA	KONULAR
Hafta 1	Yazılım Mühendisliğine Giriş
Hafta 2	Yazılım Geliştirme Süreç Modelleri
Hafta 3	Yazılım Gereksinim Mühendisliği
Hafta 4	Yazılım Mimarisi
Hafta 5	Nesneye Yönelik Analiz ve Tasarım
Hafta 6	Laboratuar Çalışması: UML Modelleme Araçları
Hafta 7	Yazılım Test Teknikleri
Hafta 8	Ara Sınav
Hafta 9	Yazılım Kalite Yönetimi
Hafta 10	Yazılım Bakımı - Yeniden Kullanımı ve Konfigürasyon Yönetimi
Hafta 11	Yazılım Proje Yönetimi (Yazılım Ölçümü ve Yazılım Proje Maliyet Tahmin Yöntemleri)
Hafta 12	Yazılım Proje Yönetimi (Yazılım Risk Yönetimi)
Hafta 13	Çevik Yazılım Geliştirme Süreç Modelleri
Hafta 14	Yazılım Süreci İyileştirme, Yeterlilik Modeli (CMM)

Kaynaklar



7.
Yazılım Sınama

Giriş

BÖLÜM HEDEFLERİ

- Yazılım Kalite değerlendirmede incelemenin (inspection) yerini anlama
- Test ve İncelemenin farkını anlama
- İnceleme ekibi yapısı ve rollerini öğrenme
- Teknik İnceleme Raporu

HAFTA İÇERİĞİ

- Yazılım inceleme temel kavramlar.
- Yazılım incelemenin önemi
- Genel yazılım inceleme süreci
- Fagan İncelemesi
- Diğer İncelemeler
- İnceleme ekibi
- İnceleme özet raporu

SİSTEM TESTİ

- Bilgisayar sistemi, donanım ve yazılım alt sistemlerinden oluşmaktadır. Bu nedenle, yazılım alt sisteminin kendi başına sınanması yeterli olmayıp, bilgisayar sistemi içerisinde de denetlenmelidir.
- Sistem testinin amacı; sistemin bütün öğelerinin uygun olarak bir araya getirildiğinin ve her birinin işlevini tam olarak gerçekleştirebildiğinin onaylanmasıdır. Yazılıma dayalı sistemlerde sistem testi;

1. düzeltme testi,
2. güvenlik testi,
3. dayanıklılık testi,
4. yetenek testi

biçimlerinde uygulanmaktadır.

SİSTEM TESTİ

- Düzeltme Testi: yazılımı çeşitli yollarla hata yapmaya zorlamak ve bu hataların düzeltilebildiğini göstermektir. Düzeltme, otomatik olarak sistem tarafından ya da işletici eliyle yapılabilmektedir.
- Güvenlik testi: sistemin zararlı dış müdahalelerden ve bilgi hırsızlığından korunabildiğinin kanıtlanmasıdır.
- Dayanıklık (stres) Testi; sistemin miktar, frekans ya da hacım bakımından anormal biçimde yüklenmesi hallerindeki dayanıklığını ölçmek amacı ile düzenlenmektedir.

SİSTEM TESTİ

- **Yetenek (performance) Testi;** gerçek zamanlı ve gömülü sistemlerde, yazılım işlem süresinin bilgisayara dayalı sistem ile uyarlığını sınamaktadır. Yeteneğin sınanması, her test basamağında uygulanmaktadır. Performans testinde gerçekleştirilen işlemler,
- performans seviyelerinin belirlenmesi
- belirli bir yük altındaki sistem çalışma zamanı performansının ölçülmesi
- sistem ayarlamaları (tuning) dir.

SİSTEM TESTİ

Performans testi aşağıdaki sorulara cevap vermektedir:

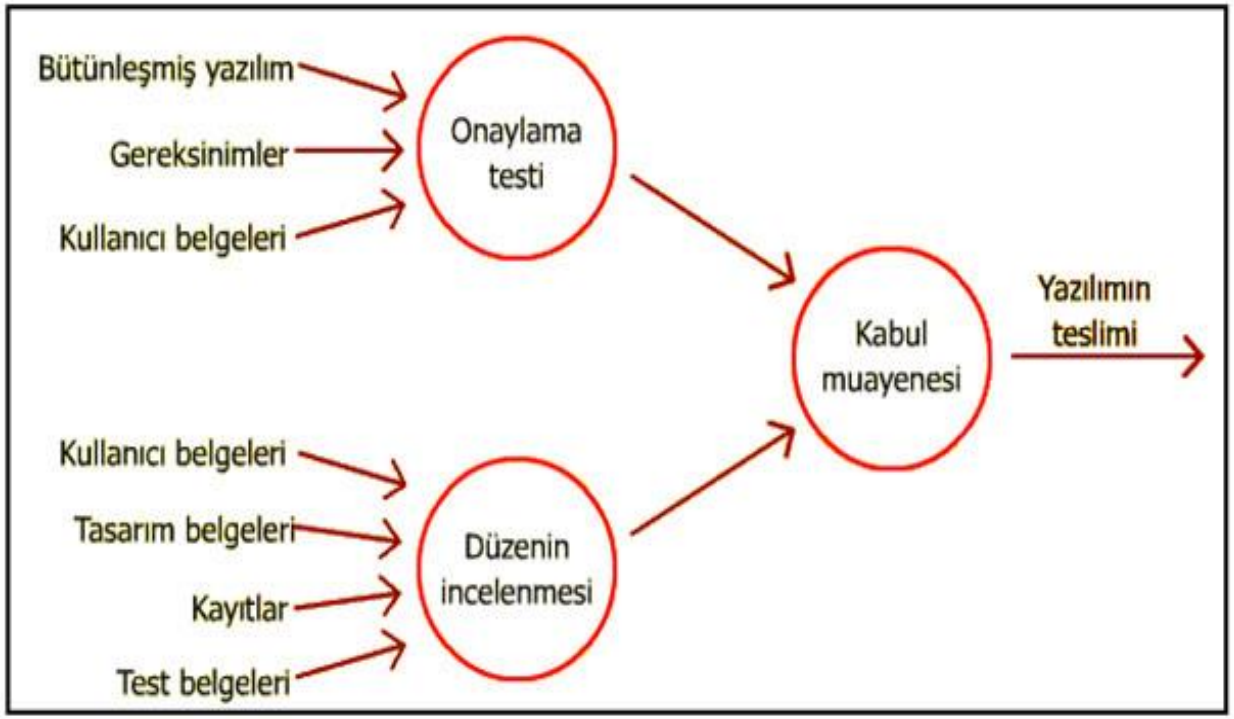
- Sistem gereksinimleri karşılıyor mu ?
- Normal şartlar altında sistem nasıl davranıyor?
- Sistem trafiğindeki artışlar işlem süresini, fonksiyonaliteyi nasıl etkiler.
- Hangi kullanıcı seviyesinde performans problemleri yaşanır?
- Performans seviyelerindeki düşüş sistemin hangi bileşeninden kaynaklanır?

SİSTEM TESTİ

- Uygulamalardaki genel performans sorunları dağılımı;
- Uygulama sunucusu - Veritabanı Sunucusu - Ağ sunucusu- Web sunucusu şeklindedir.
- Performans Testi Sırasında aşağıdaki çıktılar üretilir: • Sistem istek – cevap zamanları
  - Test Durum Dokümanları
  - Sistemin dar boğazları
  - Sistem için ideal yük
  - Sistem için ideal bant genişliği

Onaylama Testi

Bütünleme testi sonunda, yazılım bir paket halinde derlenmiş, arabirim hataları bulunmuş ve düzeltilmiş olmaktadır. Bundan sonra, onaylama testi yapılmaktadır. Bu testte, yazılımın müşteri ve kullanıcı beklentilerini gerçekleştirme olanağı denetlenmektedir. Bu amaçla; onaylama testi, düzenlik testi ve kabul muayenesi olarak yürütülmektedir



Onaylama Testi

Onaylama testi; bütünleştirilmiş yazılım ve hazırlanan kullanıcı belgeleri “yazılım gereksinimleri spesifikasyonu” ile karşılaştırılarak, gerçekleştirilmektedir. Bunun için, bir dizi kara kutu testleri uygulanmaktadır. Uygulama, daha önce hazırlanmış olan test plânına göre yürütülmektedir (kabul muayene plânı). Bu aşamada bulunan hata ve eksikler belirlenmektedir. Fakat düzeltme işlemi genellikle, müşteriye danışarak yerine getirilmektedir.

Onaylama Testi

Kabul muayenesi; müşteri tarafından yazılımın tamamının kullanıcı gözü ile incelenmesidir. Bu inceleme, proje yöneticisi tarafından kullanıcı gözü ile (alpha testi) ve sonra da müşteri tarafından (beta testi) iki aşamada gerçekleştirilmektedir.

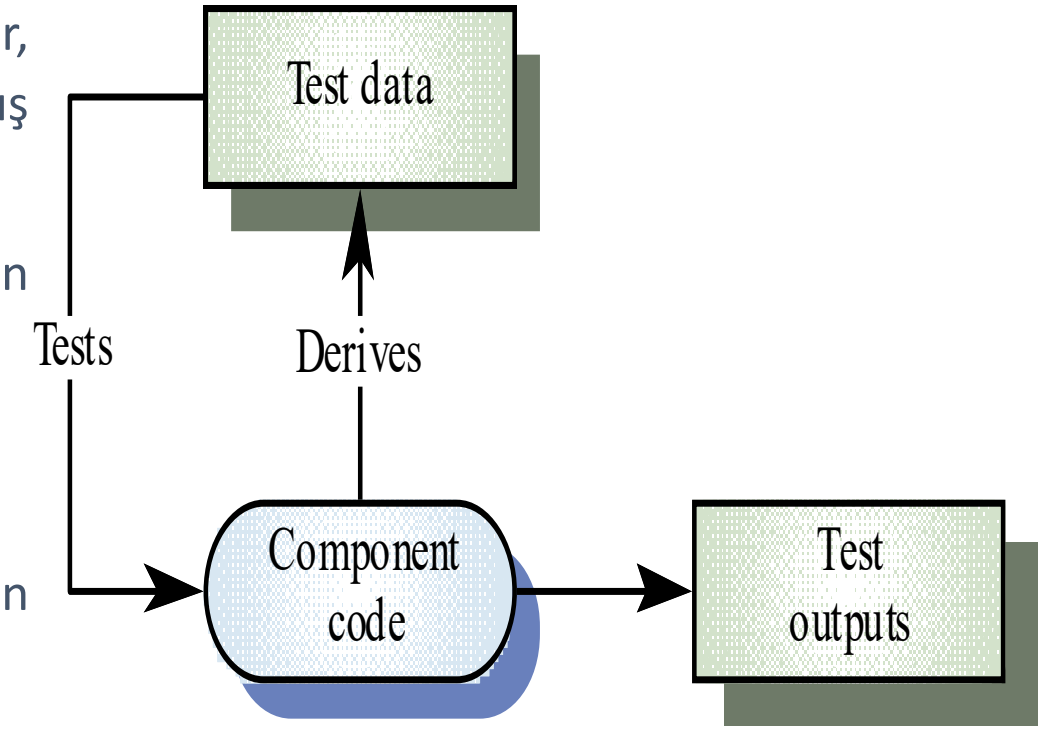
Sınama yoluyla hatalar bulunup düzeltildiği gibi, yazılımın spesifikasyonlara uygun olarak işlevlerini yerine getirdiği ve performans gereksinimlerini gerçekleştirdiği de kanıtlanmış olmaktadır. Ayrıca, bir bütün halinde yazılımın güvenirliği ve diğer kalite özellikleri de ortaya konmaktadır. Ancak, sınamalar sonunda yazılım bütün hata ve eksiklerden arındırılmış olmamakta, yine de bulunamamış ve düzeltilememiş hatalar içerebilmektedir.

TEST TEKNİKLERİ

- Saydam Kutu Testi
- Kara Kutu Testi

Saydam Kutu Testi

- Projenin kaynak ve derlenmiş kodunun sınanmasıdır. Yazılımların kaynak kodundaki akış denetimleri, koşullar, hata yakalama mekanizmaları gibi iç özellikleri sınanmış olur.
- Saydam kutu testinde, işlemsel (procedural) tasarımın kontrol yapısı kullanılmaktadır.
- Bütün bağımsız yolların en az bir kez sınanması gerekir.
- Bütün mantıksal karar noktalarında iki değişik karar için sınamalar yapılır.
- Bütün döngülerin sınır değerlerinde sınanması
- İç veri yapılarının denenmesi gerekir.



Saydam Kutu Testi

Bu test ile;

- Bir modüldeki bütün bağımsız yolların en az bir kez çalışacağı garanti edilmekte,
- Bütün mantıksal kararların "doğru" ve "yanlış" durumları denenmiş olmakta,
- Bütün döngülerin kendi içinde ve çevresinde işlerliği sağlanmakta,
- İç veri yapıları denenerek, geçerliliği güvence altına alınmaktadır.

Saydam kutu testinin uygulanmasında, temel yol testi ve döngü testi teknikleri kullanılmaktadır.

Temel Yollar Testi

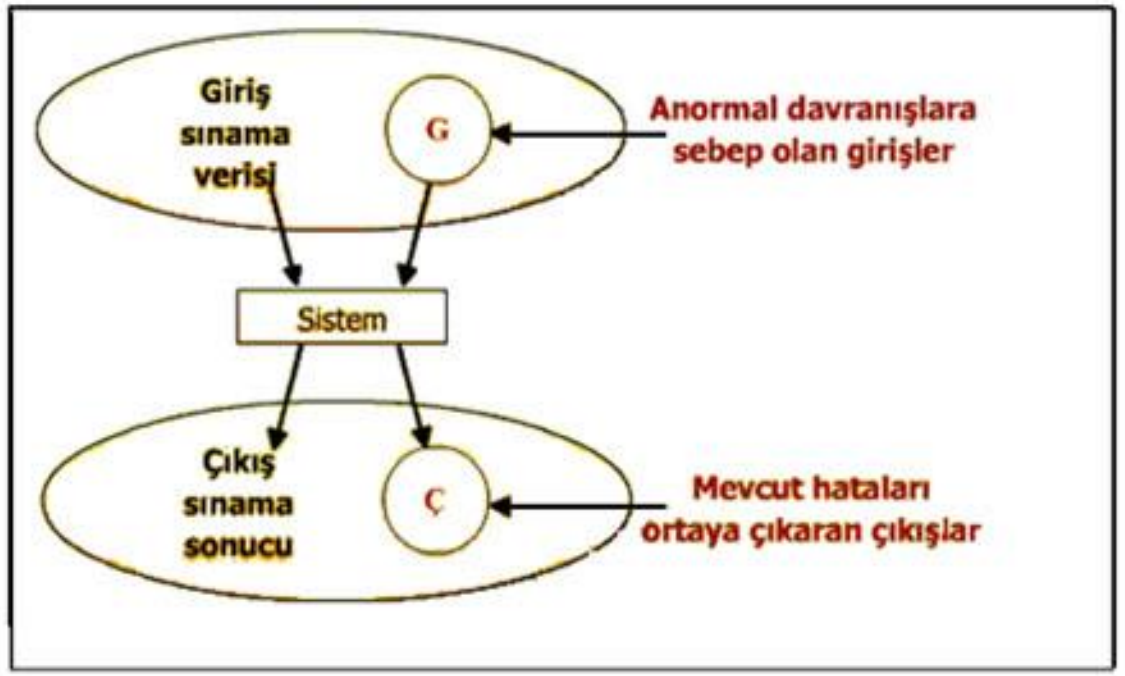
Temel yollar testi, işlemsel tasarımın mantıksal karmaşıklığını ölçmek ve bu ölçüye göre uygulama yolları için bir temel grup oluşturmak esasına dayanmaktadır. Bu grubu denemek için bir test programı düzenlemektedir. Test programları, test sırasında programdaki her deyimi en az bir kez uygulayarak denemektedir.

Döngü Testi

Döngü testi; bir saydam kutu testi olup, temel yollar analizine ek olarak yürütülmektedir. Bir döngü içerisinde bir giriş ve bir çıkışlı olarak soyutlanmış bulunan yollar da, birer döngü halinde ayrıca sınanmaktadır. Döngü testinin amacı; döngü içerisindeki başlama hatalarının, indeksleme ve artırma hatalarının, döngüyü sınırlama hatalarının bulunmasıdır. Test sonunda, döngü yapısının geçerliği onaylanmış olmaktadır. Döngüler; basit, iç içe yuvalanmış, birbirine bağlı, yapısal olmayan, olarak dört ayrı biçimde olabilmektedir.

Kara Kutu Testi

- Kara kutu sınamaları; yazılımın iç özellikleri hakkında bilgi sahibi olmaya gerek duymadan, ona dışarıdan kapalı bir kutu gibi bakıp, gereksinimlere cevap verip vermediğinin sınanmasıdır.
- Kara kutu testi; yazılımın bütünlenmesi sırasında uygulanan ve yazılım arabirimi üzerinde yapılan bir sınamadır.



Kara Kutu Testi

- Bu sınama ile, yazılım işlevlerinin yerine getirildiği, girdilerin kabul edildiği, çıktıların doğru olarak bütünlüğün sağlandığı gösterilmektedir.
- Kara kutu testinde yazılımın mantıksal iç yapısından çok, temel sistem modeli denenmiş olmaktadır. Bu nedenle, kara ve saydam kutu testleri birlikte uygulanarak, yazılım arabiriminin geçerliği onaylanmakta ve yazılımın iç işlerliğinin doğruluğu kısmen güvence altına alınmaktadır

Kara Kutu Testi

- Kara kutu testi ile;
  - Hatalı ve eksik olan işlevler,
  - Arabirim hataları,
  - Veri yapılarında ve veri tabanı erişimindeki hatalar,
  - Performans hataları,
  - Başlama ve bitirme hataları

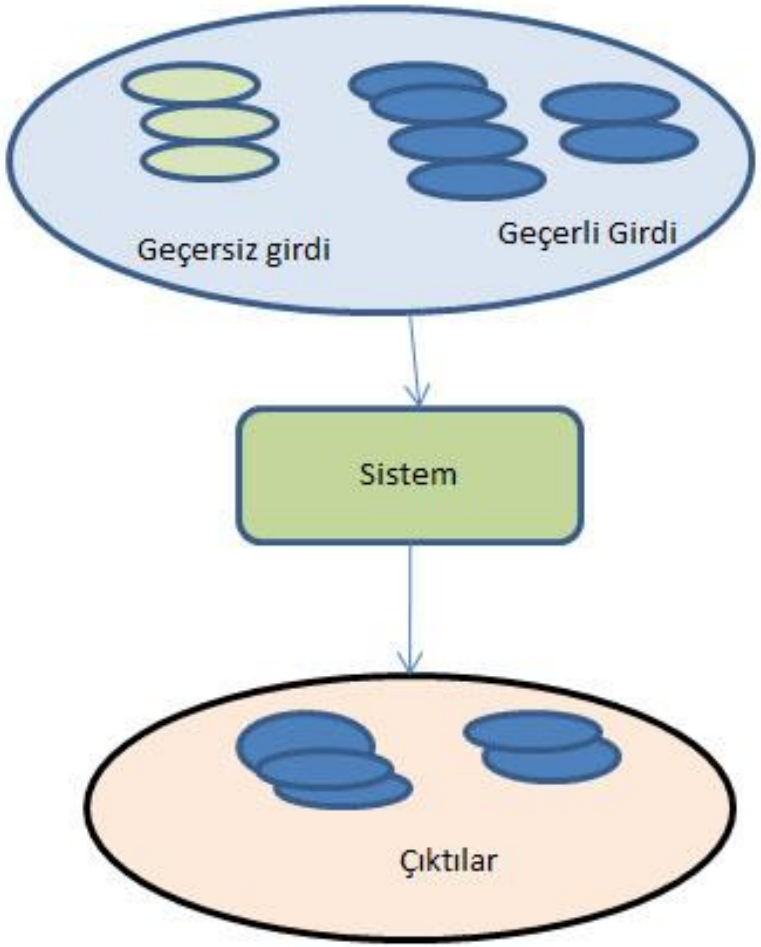
Kara kutu testi tekniklerinde de bir dizi test programları (test cases) oluşturulmaktadır.

Kara Kutu Testi

- Kara kutu testi ile, sistemin dış spesifikasyonları ve gereksinimleri sınanmaktadır. Burada, sistem düzeyinde elde edilen sonuçların kanıtlanması esas alınmaktadır. Sistemin nasıl geliştirildiği ve programdaki bütün deyimlerin işlerliği üzerinde durulmamaktadır.
- Başlıca yöntemler;
  a) eşdeğerli bölümleme,
  b) sınır değer analizi,
  c) neden-sonuç grafı çizimi,
  d) veri onaylama testi
  olarak sayılmaktadır.

Kara Kutu Testi:Eşdeğerli Bölümleme (Equıvalence Partıtıonıng) Yöntemi

- Eşdeğerli bölümleme (equivalence partitioning) yönteminde; programın girdi alanı, test programları oluşturulabilecek veri sınıflarına bölünmektedir. Böylece her test programı belirli sınıftaki hataları ortaya çıkarmakta ve daha az sayıda test programı ile yetinilmektedir.

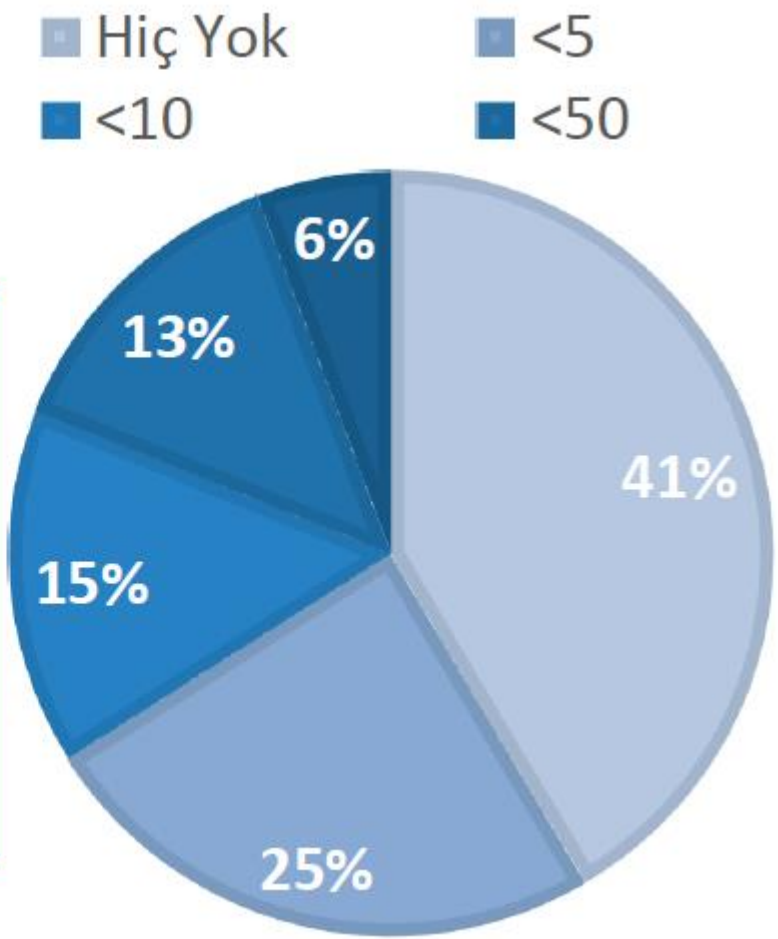


Kara Kutu Testi:Sınır Değer Analizi

- Sınır değer analizi, test verisinin; girdi alanı (ya da çıktı serisi) sınıflarının, veri yapılarının, altprogram parametrelerinin vb. sınırlarından veya uç değerlerinden seçilmesi tekniğidir.
- Genellikle en büyük-en küçük veya en az beklenen değerler ve parametreler seçilmektedir.
- Hataların genel olarak merkezden çok, kenarlarda toplandığı görülmektedir.
- Bu nedenle, sınır değerlerini denemeye yönelik test programları da geliştirilmiştir. Bu programlar "girdi" alanında olduğu kadar, "çıktı" alanında da uygulanmaktadır. Böylece, eşdeğer bölümleme tekniğini tamamlayıcı nitelikte bulunmaktadır.

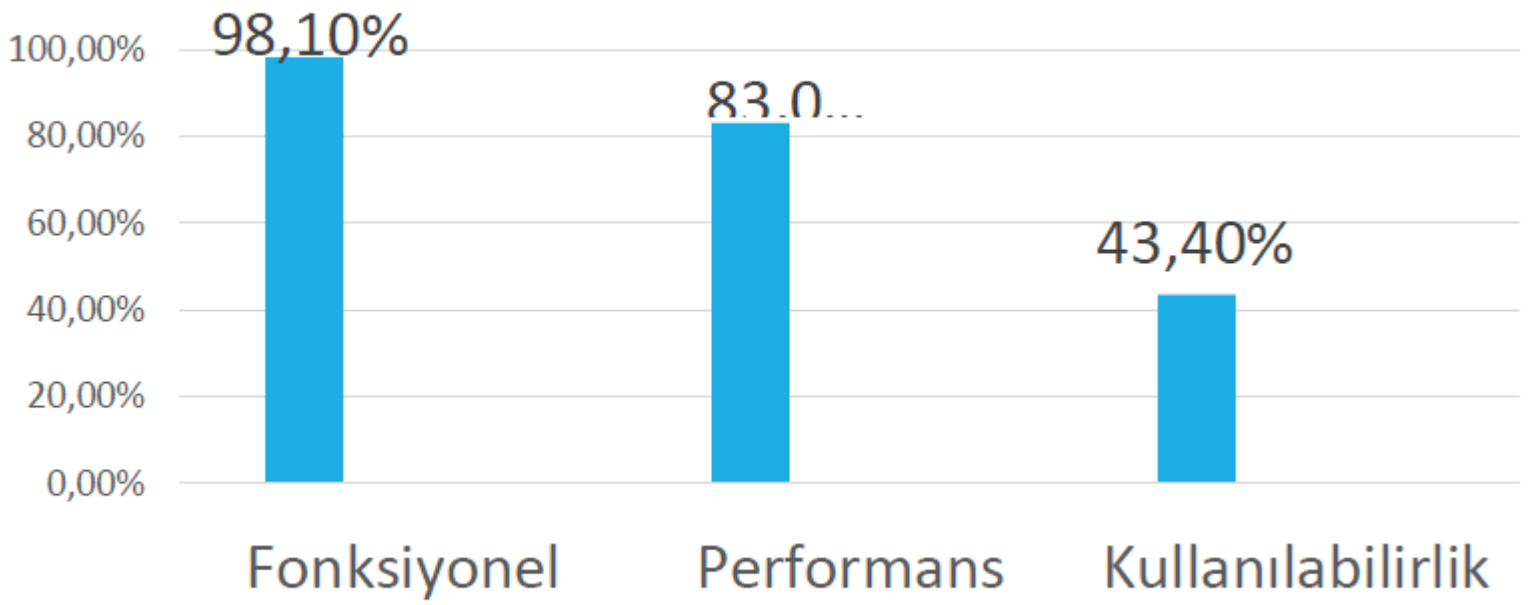
Türkiye Yazılım Kalite Raporu

- İşletmenizde kaç adet test elemanı var?



Türkiye Yazılım Kalite Raporu

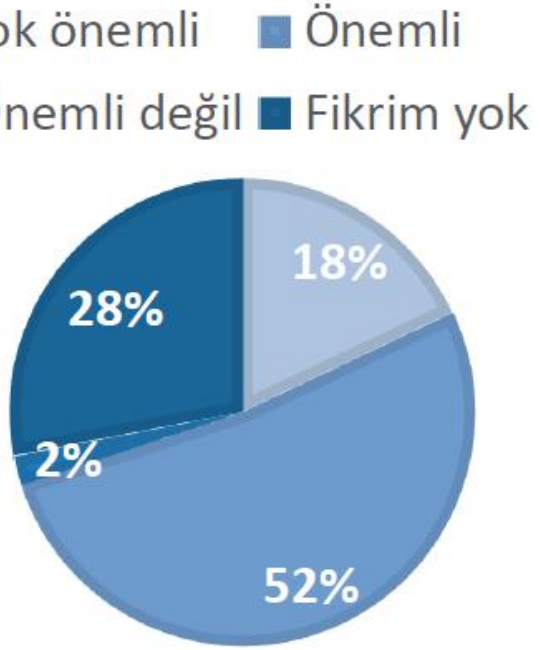
- En çok gerçekleştirdiğiniz test türleri nelerdir?



Soruya birden fazla cevap verebilme imkanı vardır.

Türkiye Yazılım Kalite Raporu

- Çalışanlarınızın test sertifikasına sahip olmaları ne kadar önemlidir?



Giriş

BÖLÜM HEDEFLERİ

- Dinamik Geçerleme (verification), yazılım test sürecini tanımlama
- Birim test ve Bütünlük test işlemlerini özetlenmesi
- Regresyon testini tanıma
- Saydam kutu Kara kutu Test tiplerini inceleme
- Performans, Dayanıklılık ve Güvenlik Testi olarak Sistem Testini tanımlama

HAFTA İÇERİĞİ

- Yazılım Test teknikleri
- Hata giderme (debuging)

TEST TEKNİKLERİ

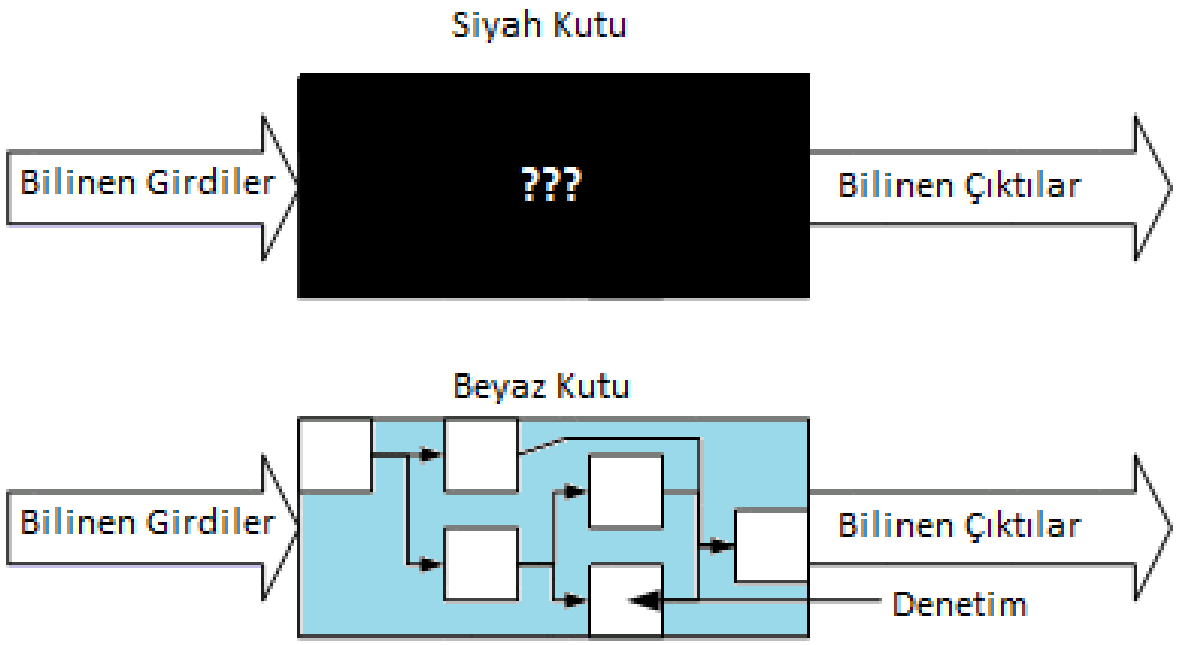
- Saydam Kutu Testi
- Kara Kutu Testi

TEST TEKNİKLERİ

- Sistemin olası tüm gereksinimleri sağlandığından emin olunması amacıyla verimli test senaryoları tasarlanması ve oluşturulması adımı, yazılım ve donanım testinin temel fazlarından biridir[1].
- Hatasız bir yazılım/donanım sistemi elde etmek pratikte imkansızdır. Bu karmaşık sistemler için de geçerlidir. İstenen kalite seviyesine ulaşmak için mümkün olan en fazla problemi ortaya çıkarma iyi bir testin amacıdır.

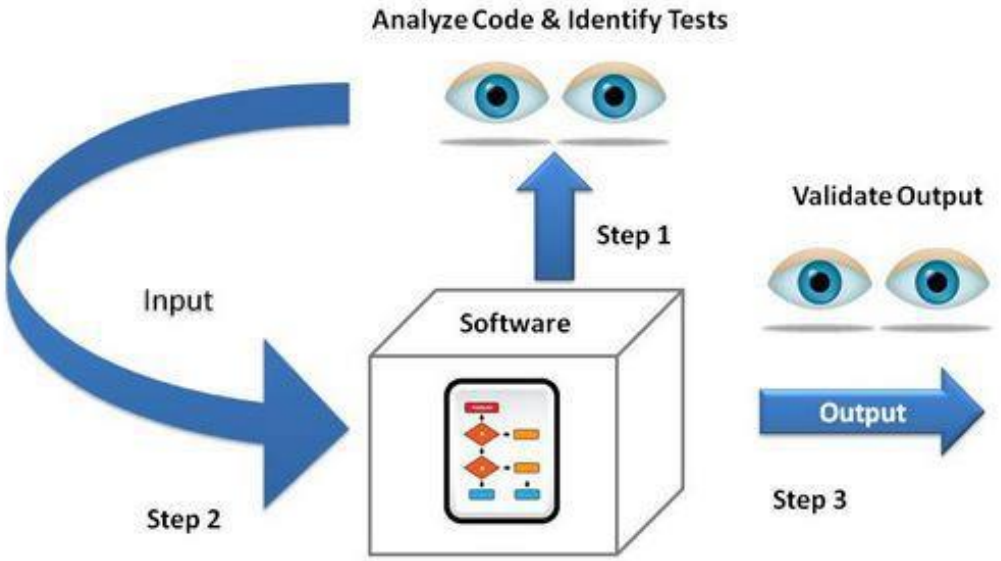
TEST TEKNİKLERİ

- Yazılım testi, yanlış ya da hata bulunması amacıyla programların ya da uygulamaların test edilmesi sürecidir.
- Genelde 2 ana gruba ayrılan metotlardan oluşmakta olup; bunlar kara kutu ve beyaz kutu (saydam kutu) testi olarak adlandırılmaktadır. Test vakalarının yürütülmesinin temel amacı test edilen yazılım için üretilen etkilerin analiz edilmesidir[1].



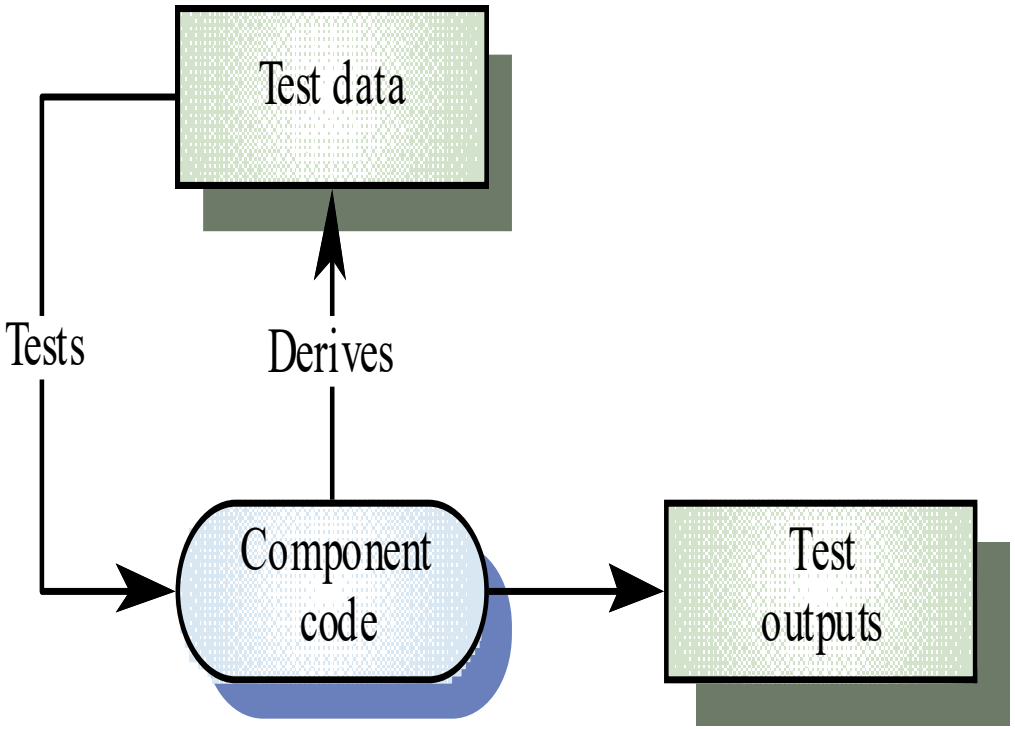
Saydam Kutu Testi

- Saydam kutu testi, modül düzeyinde uygulanmakta ve programın iç yapısını denetlemektedir.
- Test programı ile yazılımdaki koşul ve döngü durumları çalıştırılarak, mantıksal yollar sınanmaktadır. Programın tasarlandığı ve beklendiği yapıda olup olmadığını sınamak için, değişik noktalarında inceleme yapılmaktadır.
- Fakat bu yöntemde, programın tamamını sınamak olanaksızdır. Sadece sınırlı sayıdaki çok önemli mantıksal yollar seçilmekte ve bu yolların sınanması ile yetinilmektedir.



Saydam Kutu Testi

- Projenin kaynak ve derlenmiş kodunun sınanmasıdır. Yazılımların kaynak kodundaki akış denetimleri, koşullar, hata yakalama mekanizmaları gibi iç özellikleri sınanmış olur.
- Saydam kutu testinde, işlemsel (procedural) tasarımın kontrol yapısı kullanılmaktadır.
- Bütün bağımsız yolların en az bir kez sınanması gerekir.
- Bütün mantıksal karar noktalarında iki değişik karar için sınamalar yapılır.
- Bütün döngülerin sınır değerlerinde sınanması
- İç veri yapılarının denenmesi gerekir.



Saydam Kutu Testi

AVANTAJLARI	DEZAVANTAJLARI
➢ Kodun optimize edilmesine yardım eder. ➢ Gizli hatalara sebep olabilecek gereksiz satırlar kaldırılabilir. ➢ Test uzmanının kod hakkında bilgili olması sebebiyle, test senaryosunun kapsamı çok geniştir.	➢ Yetenekli bir test uzmanı gerektirdiği için maliyet artar. ➢ Gizli hataları bulmak için her uç noktaya bakmak mümkün olmadığı zaman ufak problemler ortaya çıkabilir. ➢ Kod analizcisi ve hata ayıklayıcı gibi bazı özel araçların kullanımını gerektirir.

Saydam Kutu Testi

Bu test ile;

- Bir modüldeki bütün bağımsız yolların en az bir kez çalışacağı garanti edilmekte,
- Bütün mantıksal kararların "doğru" ve "yanlış" durumları denenmiş olmakta,
- Bütün döngülerin kendi içinde ve çevresinde işlerliği sağlanmakta,
- İç veri yapıları denenerek, geçerliliği güvence altına alınmaktadır.

Saydam kutu testinin uygulanmasında, temel yol testi ve döngü testi teknikleri kullanılmaktadır.

Temel Yollar Testi

- Programcıların, test için doğrusal bağımsız yolların belirlenmesi amacıyla kullanılan bir metottur. Bu sayede yürütme esnasında mümkün olan en fazla seçenekteki test yollarını içerecek şekilde sistemin test edilmesi sağlanır. Bu metot yazılımın akış grafiğini birleştirmek için kaynak kodundan yararlanır.
- Temel yollar testi, işlemsel tasarımın mantıksal karmaşıklığını ölçmek ve bu ölçüye göre uygulama yolları için bir temel grup oluşturmak esasına dayanmaktadır. Bu grubu denemek için bir test programı düzenlemektedir. Test programları, test sırasında programdaki her deyimi en az bir kez uygulayarak denemektedir.

Temel Yollar Testi

- Grup yolu testi, ayrıntılı işlemsel tasarım ya da kaynak program üzerinde uygulanmaktadır. Uygulama;
  - Ayrıntılı tasarım veya kaynak programa dayanarak, bir akış grafı çizmek .
  - Akış grafı üzerinde döngüsel karmaşıklık (McCABE-Cyclomatic complexity) ölçüsünü saptamak, *
  - Doğrusal bağımsız yolların temel grubunu ve düğümleri belirlemek,
  - Temel grubun içerdiği her yolun denenmesi için birer test programı düzenlemek,
  - Her test programını uygulamak ve beklenen sonuç ile karşılaştırmak

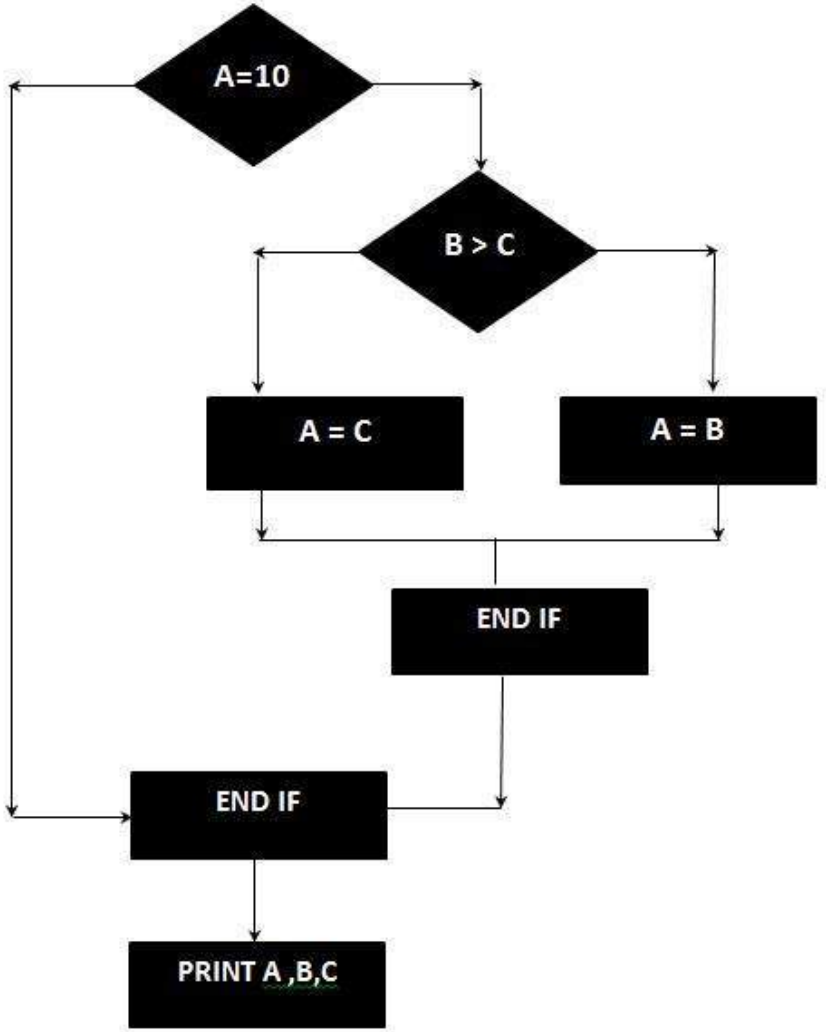
şeklinde, basamaklar halinde yürütülmektedir. Test programlarının hepsi olumlu sonuç verirse, programdaki bütün deyimlerin en az bir kez denenmiş olduğu kabul edilmektedir.

*Cyclomatic complexityy

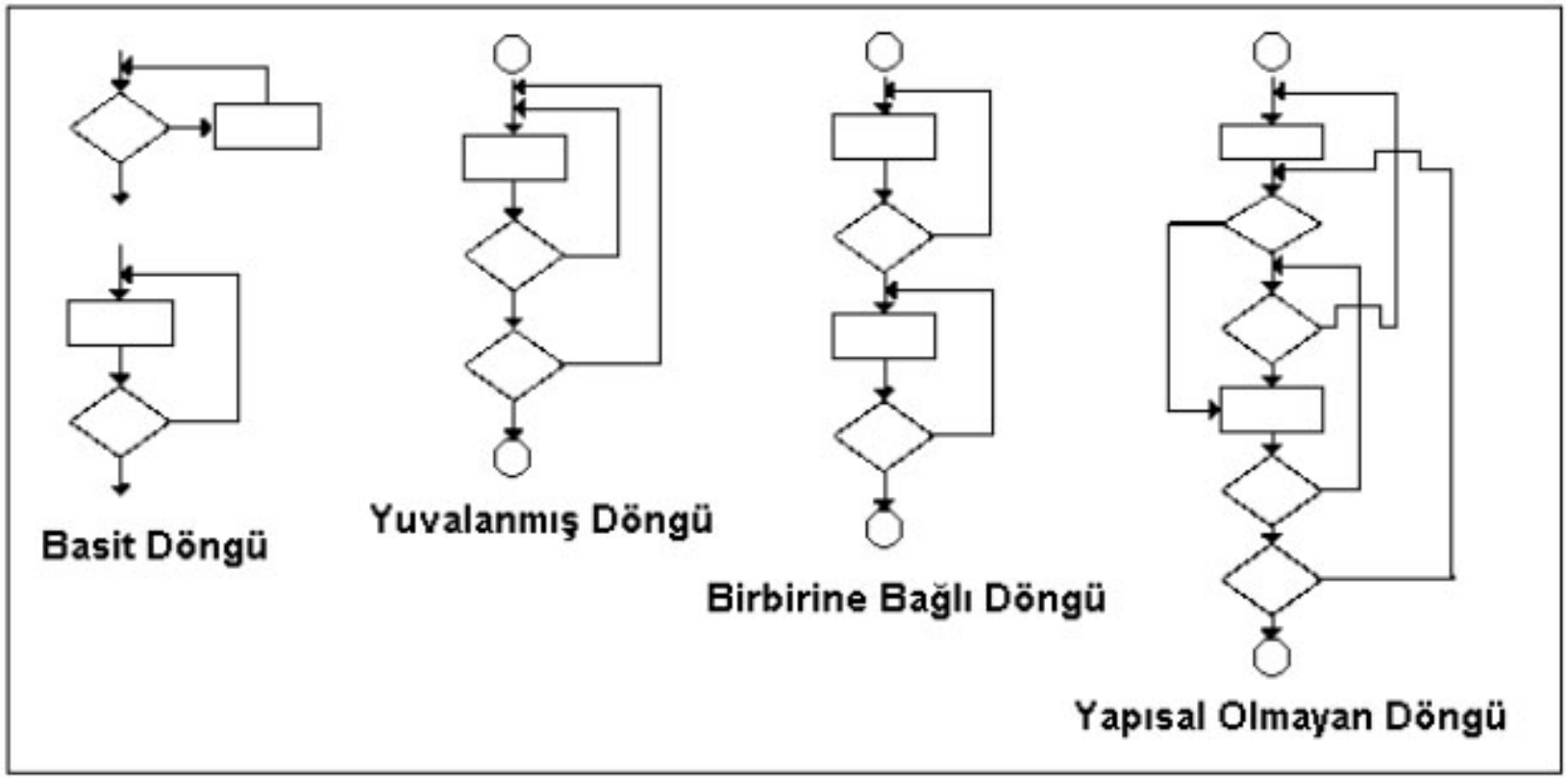
```
Cyclomatic complexity = E - N + 2*P
where,
  E = number of edges in the flow graph.
  N = number of nodes in the flow graph.
  P = number of nodes that have exit points
```

C=E-N+2*P=8-7+2*1=3

CYCLOMATIC COMPLEXITY	RISK
1 – 10	Basit ,risk az
11 – 20	daha komplex,orta derece riskli
21 – 50	kompleks,yüksek derece riskli
> 50	test edilemeyen,çok yüksek riskli(



*Cyclomatic complexityy



Ortalama alma yöntemine ait akış grafı, (R2, .. R5 , graf içindeki kapalı bölgeler, R1, graf dışında kalan bölge

Kontrol Yapı Testi

Kontrol yapı testi; 3 adet yazılım test bileşeninden meydana gelmektedir:

- **Koşul testi:** Kaynak kodunu analiz eden ve yazılım yürütmesinde varsayılan tüm koşulları dikkat alan bir metottur.
- **Veri akış testi:** Yazılımdaki tanımların ve değişkenlerin kullanımına göre test senaryolarının seçildiği bir metottur.

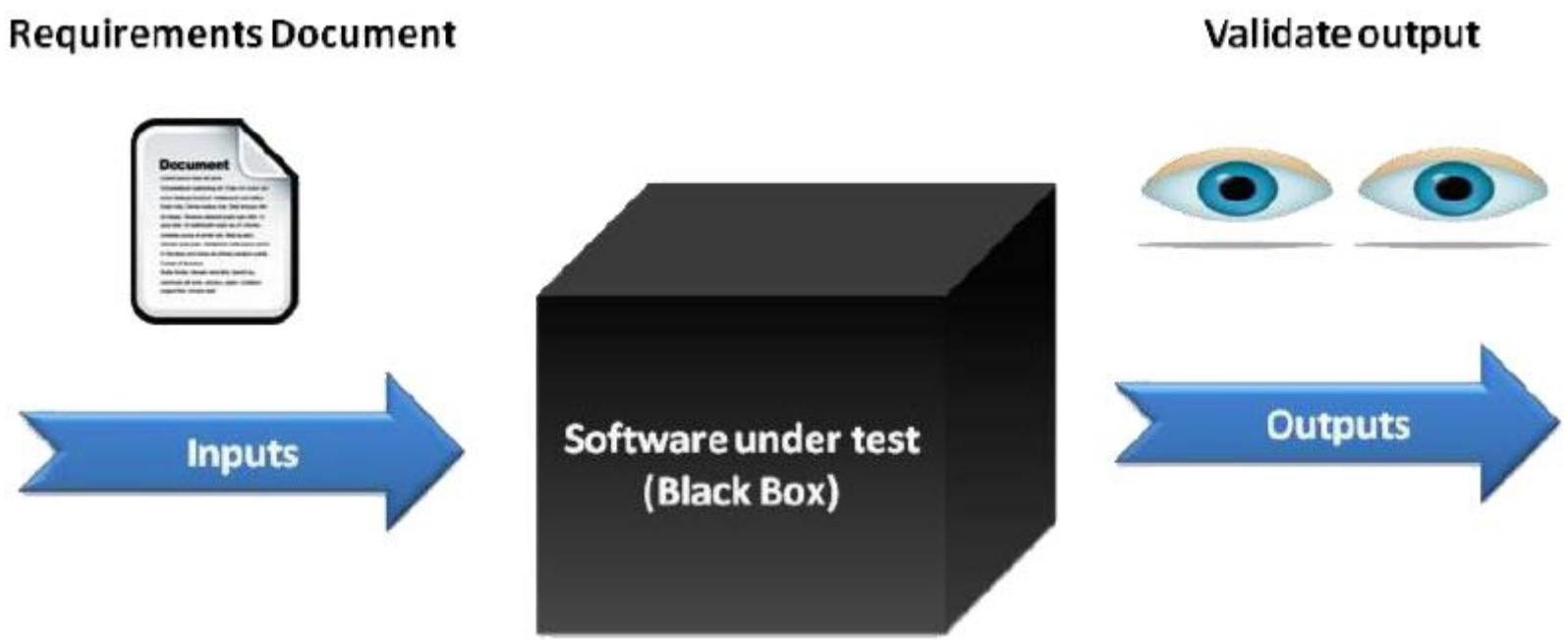
Kontrol Yapı Testi

Döngü testi: Döngüleri temsil eden bir yığın özel kodu test eden bir metottur. Genel algoritma döngüler içerdiğinden ve birçok bug bunların içinde tespit edildiğinden çok kullanışlıdır. Döngüler; basit döngüler, bağlanmış döngüler, birbirinin içine yerleştirilmiş döngüler ve yapısız döngüler gibi 4 sınıfta tanımlanmaktadır.

Döngü testi; bir saydam kutu testi olup, temel yollar analizine ek olarak yürütülmektedir. Bir döngü içerisinde bir giriş ve bir çıkışlı olarak soyutlanmış bulunan yollar da, birer döngü halinde ayrıca sınanmaktadır. Döngü testinin amacı; döngü içerisindeki başlama hatalarının, indeksleme ve artırma hatalarının, döngüyü sınırlama hatalarının bulunmasıdır. Test sonunda, döngü yapısının geçerliği onaylanmış olmaktadır. Döngüler; basit, iç içe yuvalanmış, birbirine bağlı, yapısal olmayan, olarak dört ayrı biçimde olabilmektedir.

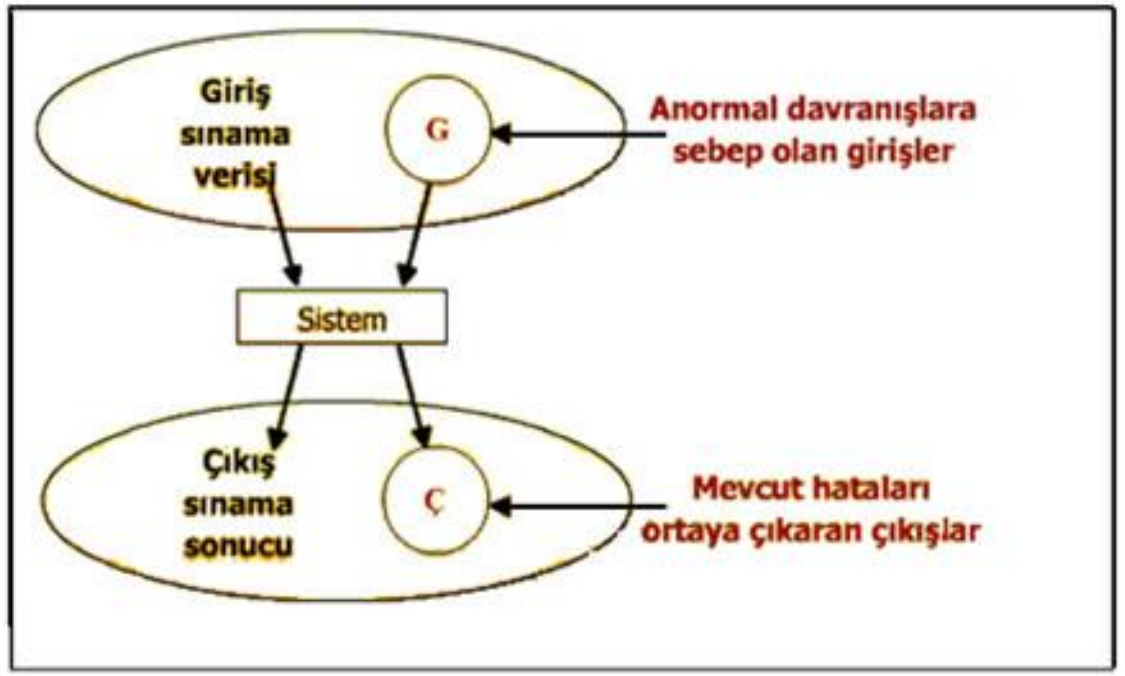
Kara Kutu Testi

- Kara kutu testleri (fonksiyonel test) bir sistemin ya da bileşenin iç mekanizmasını göz ardı ederek, sadece seçilen girdi ve yürütme koşullarına karşılık üretilen çıktılara odaklanır .



Kara Kutu Testi

- Kara kutu sınamaları; yazılımın iç özellikleri hakkında bilgi sahibi olmaya gerek duymadan, ona dışarıdan kapalı bir kutu gibi bakıp, gereksinimlere cevap verip vermediğinin sınanmasıdır.
- Kara kutu testi; yazılımın bütünlenmesi sırasında uygulanan ve yazılım arabirimi üzerinde yapılan bir sınamadır.



Kara Kutu Testi

- Bu sınama ile, yazılım işlevlerinin yerine getirildiği, girdilerin kabul edildiği, çıktıların doğru olarak bütünlüğün sağlandığı gösterilmektedir.
- Kara kutu testinde yazılımın mantıksal iç yapısından çok, temel sistem modeli denenmiş olmaktadır. Bu nedenle, kara ve saydam kutu testleri birlikte uygulanarak, yazılım arabiriminin geçerliği onaylanmakta ve yazılımın iç işlerliğinin doğruluğu kısmen güvence altına alınmaktadır.
- Kara kutu testlerinin yürütülmesi fonksiyonel gereksinimler ya da yazılımın tasarım özelliklerine bağlı olarak seçilen test senaryoları aracılığıyla gerçekleşmektedir.

Kara Kutu Testi

- Kara kutu testi ile;
  - Hatalı ve eksik olan işlevler,
  - Arabirim hataları,
  - Veri yapılarında ve veri tabanı erişimindeki hatalar,
  - Performans hataları,
  - Başlama ve bitirme hataları

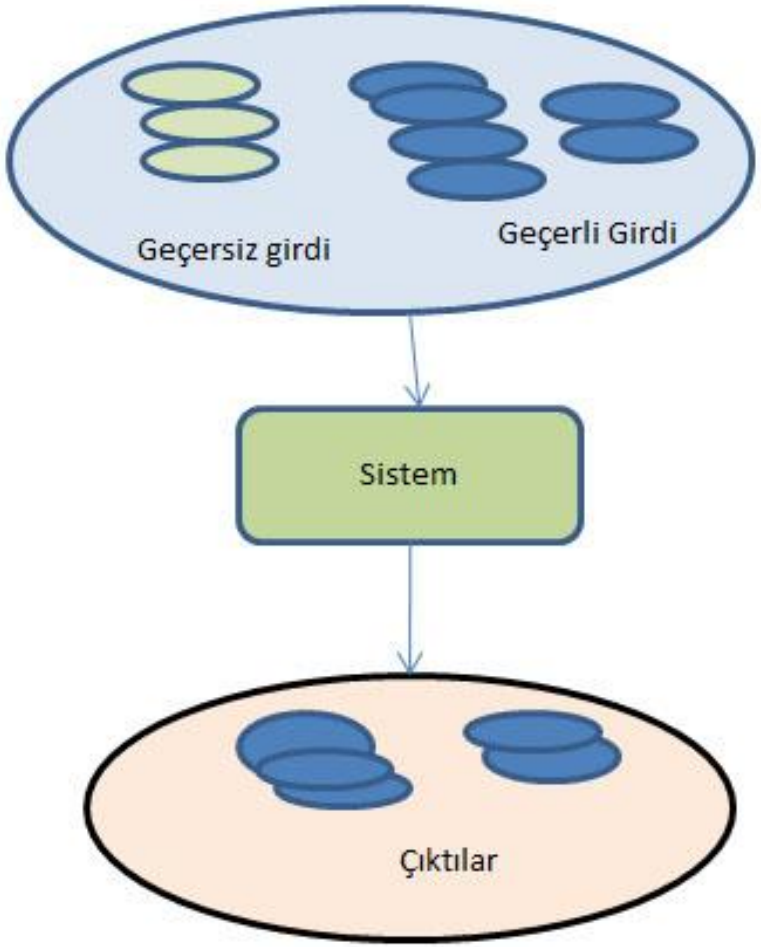
Kara kutu testi tekniklerinde de bir dizi test programları (test cases) oluşturulmaktadır.

Kara Kutu Testi

- Kara kutu testi tekniklerinde de bir dizi test programları (test cases) oluşturulmaktadır.
- Kara kutu testi ile, sistemin dış spesifikasyonları ve gereksinimleri sınanmaktadır. Burada, sistem düzeyinde elde edilen sonuçların kanıtlanması esas alınmaktadır. Sistemin nasıl geliştirildiği ve programdaki bütün deyimlerin işlerliği üzerinde durulmamaktadır.
- Başlıca yöntemler;
  a) eşdeğerli bölümleme,
  b) sınır değer analizi,
  c) neden-sonuç grafı çizimi,
  d) veri onaylama testi
  olarak sayılmaktadır.

Kara Kutu Testi:Eşdeğerli Bölümleme (Equıvalence Partıtıonıng) Yöntemi

- Eşdeğerli bölümleme (equivalence partitioning) yönteminde; programın girdi alanı, test programları oluşturulabilecek veri sınıflarına bölünmektedir. Böylece her test programı belirli sınıftaki hataları ortaya çıkarmakta ve daha az sayıda test programı ile yetinilmektedir.



Kara Kutu Testi: Sınır Değer Analizi

- Sınır değer analizi, test verisinin; girdi alanı (ya da çıktı serisi) sınıflarının, veri yapılarının, altprogram parametrelerinin vb. sınırlarından veya uç değerlerinden seçilmesi tekniğidir.
- Genellikle en büyük-en küçük veya en az beklenen değerler ve parametreler seçilmektedir.
- Hataların genel olarak merkezden çok, kenarlarda toplandığı görülmektedir.
- Bu nedenle, sınır değerlerini denemeye yönelik test programları da geliştirilmiştir. Bu programlar "girdi" alanında olduğu kadar, "çıktı" alanında da uygulanmaktadır. Böylece, eşdeğer bölümleme tekniğini tamamlayıcı nitelikte bulunmaktadır.

Test Senaryosu

Bir test senaryosu, özel bir amaç için geliştirilen (belirli bir program yolu uygulamak veya özel bir gereksinimle uygunluğun doğrulanması gibi) bir grup test girdisi, yürütme koşulları ve beklenen sonuçların olduğu bir gruptur. Test senaryoları, sistemin gereksinimler tarafından belirlenen hedef değerlerle geçmesini ya da kalmasını doğrulamak için bir yol göstermektedir.

Avantajları

- Test senaryolarının yürütülmesi ve dokümantasyonu geliştirme fazında hata bulma,
- Ortaya çıkarılabilir hata sayısını maksimize etme,
- Olgunlaşmayan ürün lansmanlarını önleme,
- Teknik destek maliyetleri azaltma,
- Özelliklere uygunluğu değerlendirme,

Test Senaryosu

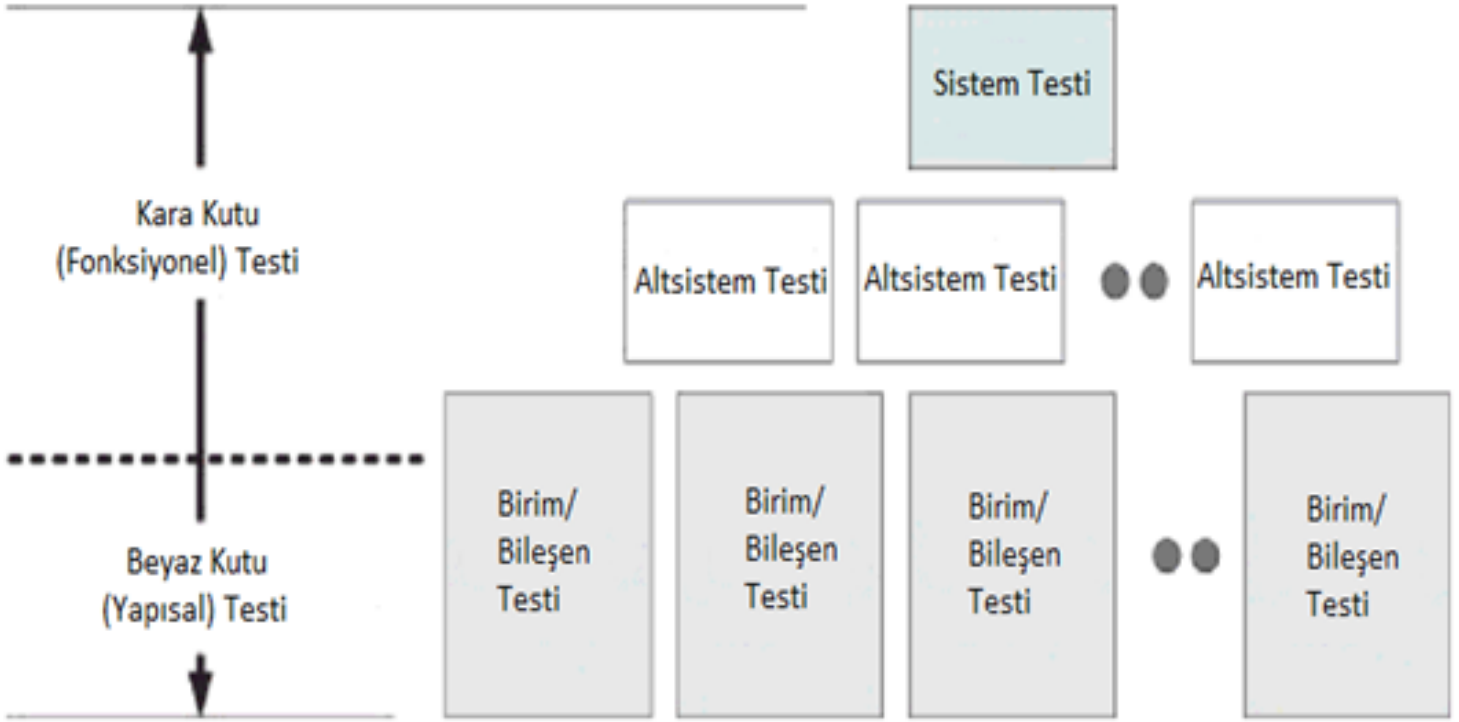
- Kurallara uyma,
- Güvenlikle ilgili riskleri azaltma,
- Hatalara rağmen ürünün kullanımı için güvenli senaryoları bulma,
- Kaliteyi değerlendirme-temin etme
- Ürünün hatasızlığını doğrulama

Kara Kutu ve Beyaz Kutu Metotları Karşılaştırılması

- Sınama sonucu saptanan hata ve eksiklerin nedenlerinin bulunup, düzeltilmesi gerekmektedir. Hataları giderme (debugging) adı verilen bu işlemde, belirtiler ile nedenlerinin karşılaştırılması, sonra da hataların düzeltilmesi yoluna gidilmektedir. Bug nedenleri büyük oranda gereksinim analizinden kaynaklanmaktadır.
- Bu iki metot arasındaki temel fark, kara kutu testlerinin daha üst seviyede yapılır olması, diğer bir değişle test eden kişinin sistemin detaylarını bilmesini gerektirmemesidir (uygulama kaynak kodu gibi). Beyaz kutu metodunda ise test ediciler sistemin her bir bileşeninin davranışını ve kaynak kodunu bilmek zorundadır. Genellikle, beyaz kutu testi her bir tekil yazılım bileşeninde gerçekleşmesi nedeniyle daha doğru ve daha tam görünmektedir.

Kara Kutu ve Beyaz Kutu Metotları Karşılaştırılması

- Kaynak kodunu kullandığı için geliştirme süreci esnasında kullanılabilen tek test tipi olması beyaz kutu testinin avantajıdır. Kara kutu testine göre dezavantajı ise son yazılımda (sistem) kullanılamıyor olmasıdır
- Şekilde hiyerarşinin belirli katmanında hangi metodun kullanılacağı ve sistem testinin yazılım mimarisine uygun olarak nasıl parçalandığı görülmektedir



Kara Kutu ve Beyaz Kutu Metotları Karşılaştırılması

SIRA	KARA KUTU	BEYAZ KUTU
1	Uygulamanın iç dinamikleri test uzmanı tarafından bilinmemektedir.	Test uzmanı uygulamanın tüm iç dinamiklerini bilmektedir.
2	Test sorumlusunun beklentilerine göre yapılmaktadır.	Tüm iç dinamikleri bilindiğinden uygun test datası hazırlanarak yapılmaktadır.
3	Zaman maliyeti en düşük olan test tekniğidir.	Zaman maliyeti en yüksek olan test tekniğidir.
4	Algoritma testi için uygun değildir.	Algoritma testi için uygundur.
5	Veri kısıtları bilinmediğinden, testler deneme yanılma yöntemi ile yapılır. Hatalar rastlantısal olarak tespit edilir.	Uygulamanın işlediği veri türleri ve uygulamanın ilgili veriyi işleme şekli bilindiğinden, testler bilinçli olarak uygun test datası hazırlanarak yapılır.

Hata Giderme

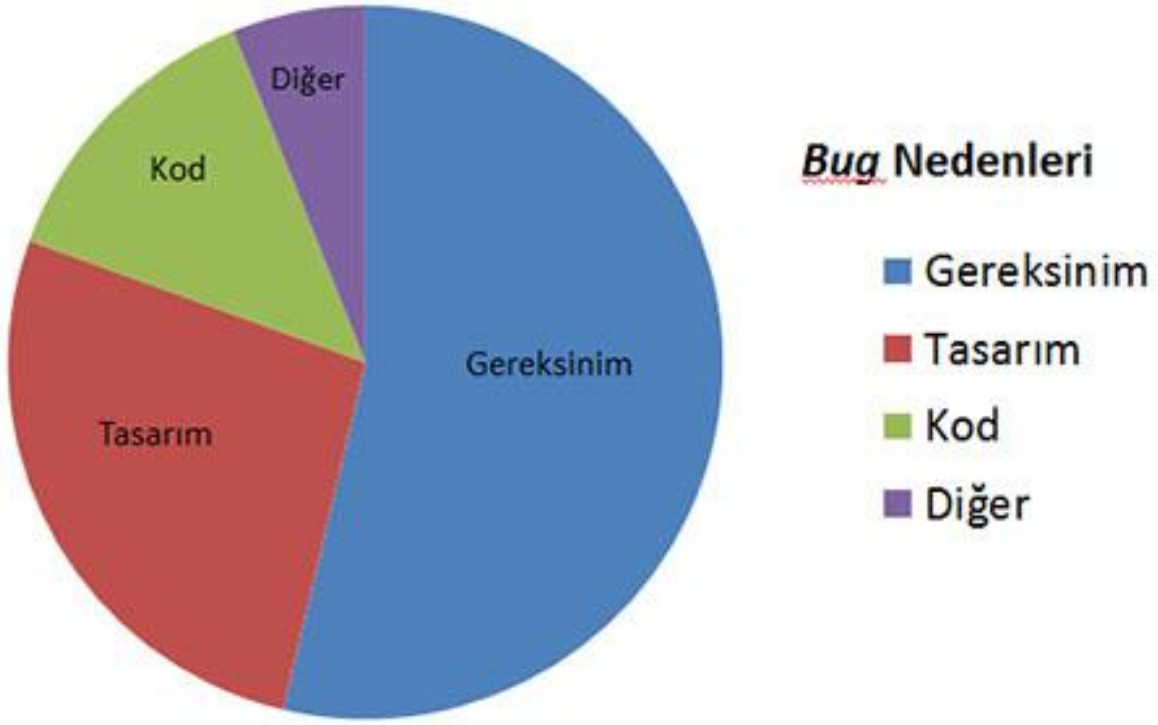
- Kullanıcının geliştirilen yazılım ile yapmak istediklerinin yazılım tarafından yapılmaması veya eksik yapılmasına hata denilir. Hatalar genellikle testler sırasında tespitedilir. Testler sırasında belirlenen yazılım hatalarının düzeltilmesi iş gücü kaybına ve maliyete neden olur. Testlerden önce, gözden geçirme faaliyetleri sırasın da hatalar bulunduğunda daha azmaliyetle düzeltilebilir. Projeiçinde hataların ve düzeltici faaliyetlerin kontrol altına alınması için bir yaşam döngüsü mevcuttur.

Hata Giderme

- İlkprogram bug'ı 1947 yılında Grace Murray Hopper'in Harvard Üniversitesi'nde kullandığı MarkII Aiken isimli röle bazlı hesaplayıcıda bulundu. 9 eylül 1947 tarihinde hesaplayıcının programlandığı şekilde çalışmadığı, sorun çıkardığı belirlendi. Yapılan araştırma üzerine bir rölenin bacakları arasında bir güvenin sıkışıp kaldığı görüldü. Program hatasının sebebi bulunmuştu; bir güve yani bir böcekti (bug).

Hata Giderme

- Sınama sonucu saptanan hata ve eksiklerin nedenlerinin bulunup, düzeltilmesi gerekmektedir. Hataları giderme (debugging) adı verilen bu işlemde, belirtiler ile nedenlerinin karşılaştırılması, sonra da hataların düzeltilmesi yoluna gidilmektedir. Şekil 5.11 de görüldüğü gibi bug nedenleri büyük oranda gereksinim analizinden kaynaklanmaktadır.

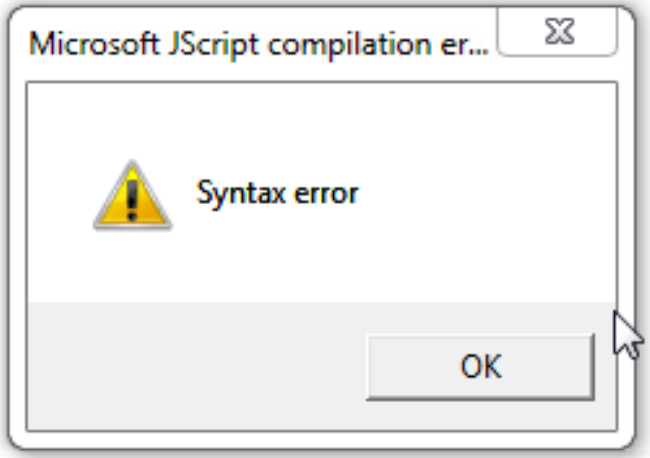


Hata Giderme

- Yazılım geliştirme sürecinin herhangi bir aşamasında (temel olarak analiz, tasarım, kodlama, test, bakım) yapılan insanı hatalardır. Hiçbir yazılımcı veya programcı isteyerek hata yapmaz. Dolayısı ile programı kendi kendine test ederken yaptığı hatayı da görmemesi normaldir. Sıfır hatalı bir yazılım üretmek pratikte mümkün değildir. Ancak doğru hata yönetimi yaparak hata sayısını azaltabilir ve hata oluştuğunda müdahale için daha hızlı olabilirsiniz. Uygulama geliştirme aşamasında hatalar 3 grupta değerlendirilir:
  - Syntax Error – Sözdizimi Hataları
  - Run-time Error – Çalışma Zamanı Hataları
  - Logic Error (Bug) – Mantıksal Hatalar (Böcek)

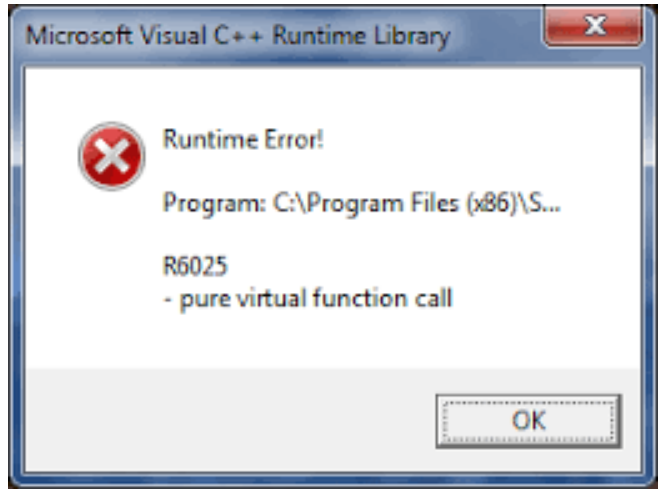
1. Syntax Error – Söz dizimi Hataları

- Yazılan programda programlama dili kurallarına aykırı bir takım ifadelerden dolayı karşılaşılabilecek hatalardır. Düzeltilmesi basit hatalardır. Hatanın bulunduğu satır derleyici tarafından rapor edilir. Günümüz IDE'lerinde bu sıkıntılar neredeyse yok denecek kadar azdır. Özelikle kod editörlerinin gelişmiş yazım denetimi sayesinde yazılımcılar söz dizimi hatalarını derlemeye gerek bile kalmadan fark edebiliyorlar. Eğer bir derlemede Syntax Error alındı ise obje kod üretilememiştir demektir.



2. Run-time Error – Çalışma Zamanı Hataları

- Programın çalıştırılması sırasında karşılaşılan hatalardır. Programcının ele almadığı bir takım aykırı durumlar ortaya çıktığında programın işletim sistemi tarafından kesilmesi ile ortaya çıkar. Bu tip hatalarda hata mesajı çoğunlukla çalışan işletim sisteminin dili ile verilir. Eğer bu tip hataları kullanıcı ele almışsa, program programcının vereceği mesajlarla ve uygun şekilde sonlandırılabilir. Bu tip hataların nerelerde ve hangi şartlarda ortaya çıkabileceğini bazen kestirmek zor olabilir. Örneğin olamayan bir dosya açmaya çalışmak, var olan bir dosyanın üzerine yazmaya çalışmak, olmayan bir bellek kaynağından bellek ayırtmaya çalışmak, olmayan bir donanıma ulaşmaya çalışmak vs.



3. Logic Error (Bug) – Mantıksal Hatalar (Böcek)

- Karşılaşabileceğiniz en tehlikeli hatadır. Programlama mantığında bir takım şeylerin yanlış düşünülmesinden kaynaklanır. Hata test aşamasında veya müşteri kullanımı sırasında ortaya çıkar. Örneğin: Hesaplanması gereken veya bulunması gereken değerlerin eksik veya yanlış hesaplanması mantıksal bir hatadır. Bu sorunun giderilebilmesi için Analiz aşamasına kadar geri dönülmesi gerekebilir. Bazen bu hatanın nereden kaynaklandığını bulabilmek çok zor olmaktadır. Gerek serbest yazılım gerek ticari yazılımların tümünde bug dediğimiz mantıksal hatalar bulunur. Çünkü hatasız program yazabilmek çok zordur. Günümüzde en etkin yazılım firmaları bile yazılımlarında bug olduğunu kabul eder ve zaman zaman bu bugları giderebilmek için ya yazılımlarına yama yazılımı (Update, Patch) üretirler ya da o yazılımın yeni bir versiyonunu piyasaya sürerler.

Hata Düzeyleri

Hataların düzeyleri,

- Ölümcül,
- Kritik,
- Büyük,
- Orta,
- Küçük ve
- Görünüm (kozmatik)

Olarak tanımlanır. Ölümcül hatalar, testin devamını engeller, kritik hatada test devam eder, ama düzeltilmeden yazılım teslim edilemez, büyük hata giderilmeden yazılım teslimi zararlıdır. Orta hata ile Testler devam edebilir ürün hata ile teslim edildiğinde telafisi mümkün sorunlar çıkartabilir. Küçük hata ile testler devam edebilir ve ürün bu hata ile teslim de edilebilir, yazılımın önemli bir sonuç doğurmaz. Görünüm hatalar ise Yazılımın, renk, font, büyüklük ile ilişkili hatalarıdır.

Hata Düzeyleri

Hataların düzeyleri,

- Ölümcül,
- Kritik,
- Büyük,
- Orta,
- Küçük ve
- Görünüm (kozmatik)

Kaynaklar

1. A. Avcıoglu, Bilgisayar Tabanlı Sistemlerde Test Otomatızasyonunun Tasarlanması Ve Gerçeklenmesi, Hacettepe Üniv. Lisansüstü Tezi, Ankara, 2015