YMT 412-Yazılım Kalite Ve Güvencesi

Test Stratejileri

Bölüm-4

Fırat Üniversitesi Yazılım Mühendisliği Bölümü

İçindekiler

1. Yazılım Testi

- Yazılımları test etmek için kullanılabilecek birçok yöntem vardır. Örneğin, sistem tamamen inşa edilene kadar beklenilir ve hataları bulma amacıyla tüm sistem üzerinde testler yapılır. Bu yaklaşım ilgi çekici olmasına rağmen çalışmaz. Tüm taraflarda memnuniyetsizlik uyandıracak hatalı bir yazılımla sonuçlanır.

- Diğer bir örnek ise, test günlük olarak, sistemin herhangi bir parçası inşa edildiğinde gerçekleştirilir. Bu yaklaşım daha az ilgi çekici olmasına rağmen çok verimlidir. Ne yazık ki birçok yazılım geliştirici, bu yöntemi kullanmakta endişe duymaktadır.

2. Test Tipleri

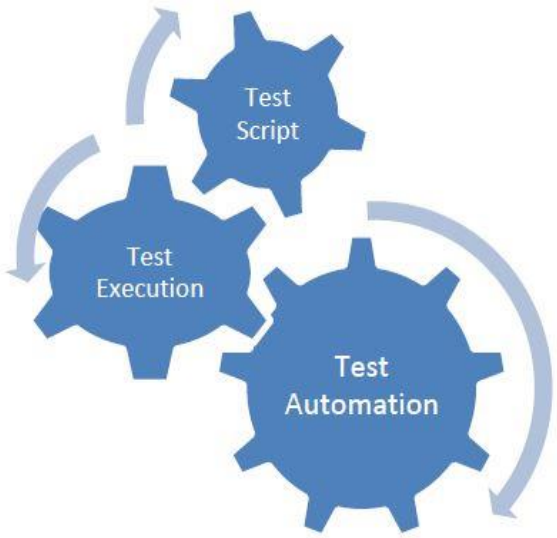
- Yazılım geliştirme yaşam döngüsü boyunca kullanılan test tipleri:
  - El ile yapılan (Manual) Testler
  - Otomasyon (Automation) Testleri

2.1. Manuel Test

- Yazılımın herhangi bir test otomasyonu veya herhangi bir script kullanılmadan el ile ile test edilmesidir. Yazılımdaki herhangi bir beklenilmeyen davranışı veya hatayı bulmak için yapılır. Küçük projelerde kullanılması daha uygundur.

2.2. Otomasyon Testi

- Test otomasyonu olarak da bilinen otomasyon testi, test yapan kişilerin scriptleri yazdığı ve yazılımı test etmek için başka yazılımlar kullandığı test türüdür. Otomasyon testi, test senaryolarının hızlı, art arda ve tekrarlarca uygulanabilmesini sağlar. Yük, performans, stress gibi çok kullanıcı gerektiren testlerde ve sık sık değişiklik yapılan regresyon testlerinde kolaylık sağlar.

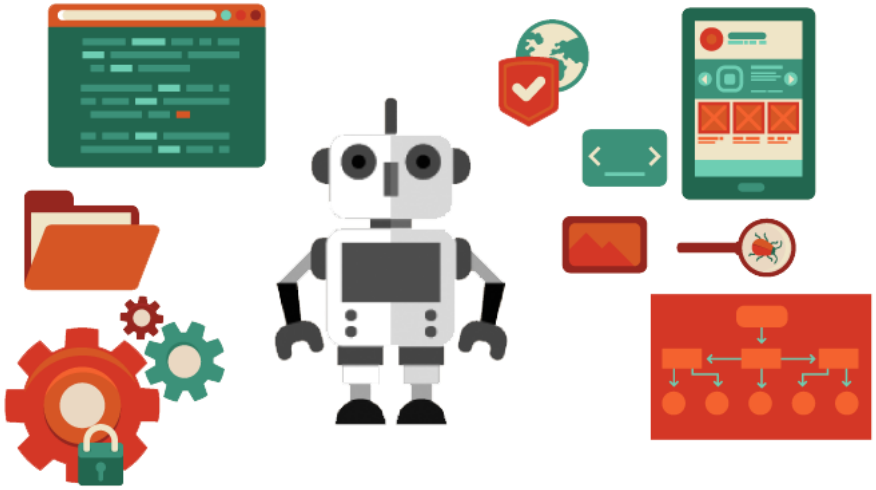


2.3. Test Tiplerinin Karşılaştırılması

Otomasyon Testi	Manuel Test
Testler daha hızlı çalıştırılır.	Otomasyon testinden daha yavaştır.
Bir çok testi defalarca çalıştırabilir.	Bir veya iki kez çalıştırılacak olan testlerde kullanılması uygundur.
Sık sık değişiklik içeren regresyon testlerinde verimli çalışır.	Regresyon testlerini manuel olarak yapmak zordur.
Karmaşık projelerde kolaylık sağlar.	Karmaşık projeler manuel olarak test yapılmaz.
Test otomasyonlarını satın almak maliyetlidir.	Daha az maliyetlidir.
Kullanıcı ara yüzü testlerinde bazen verimli olabilir.	Kullanıcı ara yüzü testlerinde çok verimlidir.
Daha doğru sonuçlar üretir.	Otomasyon testinden daha az güvenilirdir.

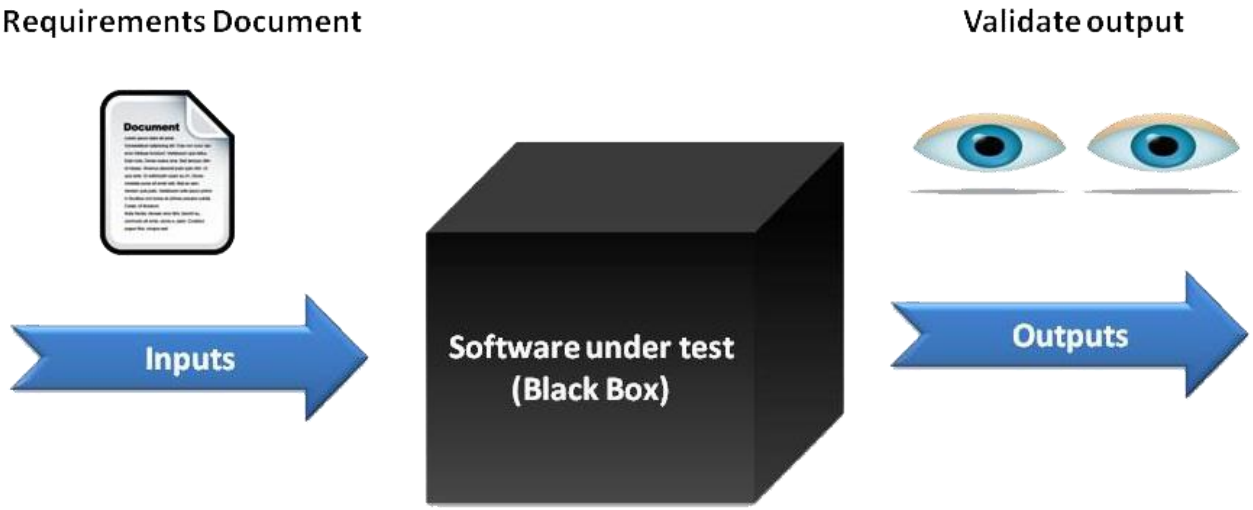
3. Test Metotları

- Yazılım testlerinde kullanılan bir çok metot vardır. Bu derste aşağıdakileri metotları inceleyeceğiz:
  - Beyaz kutu testi(White box testing)
  - Gri kutu testi (Gray box testing)
  - Kara kutu testi (Black box testing)



3.1. Kara Kutu Testi

- Uygulamanın iç yapısıyla ilgili hiçbir bilgiye sahip olmayan test tekniğidir. Test uzmanı, sistem mimarisiyle ilgilenmez ve kaynak kodlara erişmez. Kara kutu testinde test uzmanı sistemin kullanıcı ara yüzünde belirtilen girdileri sağlayarak çıktıların doğru olmasını bekler.



3.1. Kara Kutu Testi

- Kara kutu testi kullanılarak yakalanabilecek hatalar:

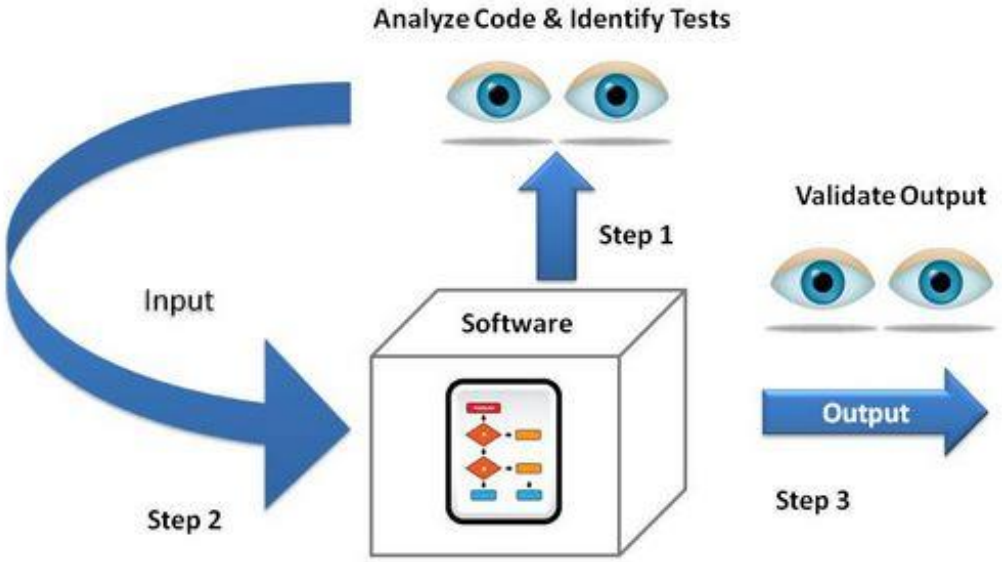
1. Doğru olmayan ya da kayıp fonksiyonlar
2. Ara yüz hataları
3. Veri yapılarındaki hatalar ya da harici veritabanı bağlantısı hataları
4. Davranış ya da performans hataları
5. Başlatma ve sonlandırma hataları

3.1. Kara Kutu Testi

AVANTAJLARI	DEZAVANTAJLARI
➢Kod erişimi gerektirmediği için daha kolaydır. ➢Birçok orta vasıflı test uzmanı, uygulamanın içeriği, programlama dili yada çalıştığı işletim sistemi hakkında bilgi sahibi olmadan uygulamayı test edebilir.	➢Test durumu tasarlamak zordur. ➢Sadece birkaç test senaryosu seçilip uygulandığı için kapsamı dardır. ➢Test uzmanı özel kod bölümlerini veya hata eğilimli alanları hedef alamadığı için kapsam yetersizdir.

3.2. Beyaz Kutu Testi

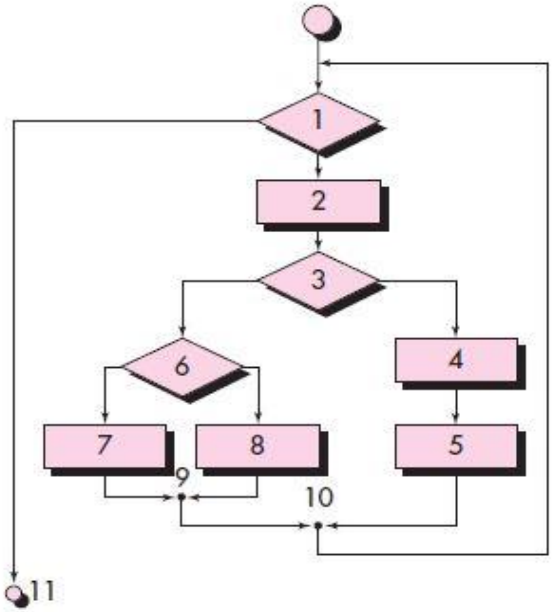
- Beyaz kutu testi kodun yapısını ve iç mantık yapılarını detaylı olarak inceler. Saydam kutu testi ya da açık kutu testi olarak da bilinir. bir uygulamada beyaz kutu testi yapmak için, test uzmanı kodun iç çalışma yapısını bilmek zorundadır.



3.2. Beyaz Kutu Testi

- Beyaz kutu testi kullanılarak yapılabilecek denetimler arasında:
  - Bütün bağımsız yolların en azından bir kere sınanması,
  - Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması,
  - Bütün döngülerin sınır değerlerinde sınanması,
  - İç veri yapılarının denenmesi

bulunur.



3.2.Beyaz 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.

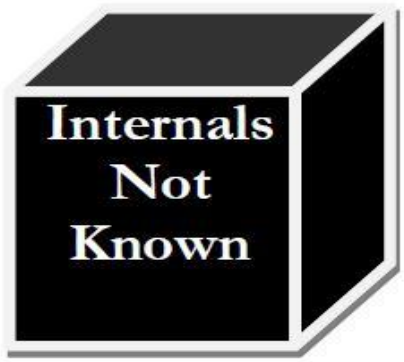
3.3. Gri Kutu Testi

- Kara kutu ve beyaz kutu testlerinin birleşimidir. Test uzmanının veri tabanına ve dokümanlara erişimi vardır. Böylece tasarıma ve verilere uygun test dokümanı üretebilir. Yani uygulamanın iç işlemlerine kısmen erişime izin verir.

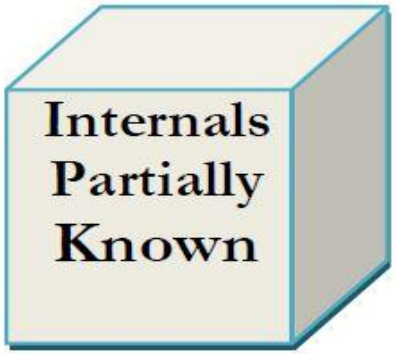
3.3.Gri Kutu Testi

AVANTAJLARI	DEZAVANTAJLARI
➢Beyaz kutu ve kara kutu testinin yararlarının birleşimini sunar. ➢Gri kutu test uzmanları, kaynak kod yerine ara yüz tanımlamaları ve fonksiyonel özellikleri kullanır. ➢Test, tasarımcı bakış açısıyla değil kullanıcı bakış açısıyla yapılır.	➢Kaynak koda erişim sınırlı olduğundan kod inceleme ve test kapsamı sınırlıdır. ➢Yazılım tasarımcısı halihazırda bir test çalıştırıyorsa gri kutu testi gereksiz olabilir. ➢Mümkün olan her giriş sisteminin test edilebilmesi gerçekçi değildir çünkü çok fazla zaman alır. Bu yüzden bazı program yolları test edilemez.

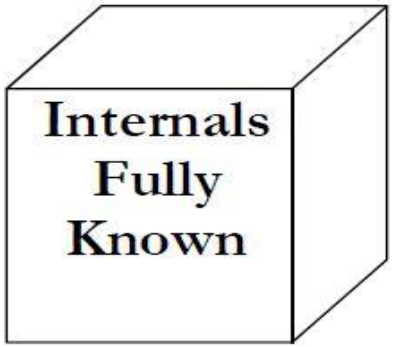
3.4. Metotların Karşılaştırılması



Kara Kutu Testi

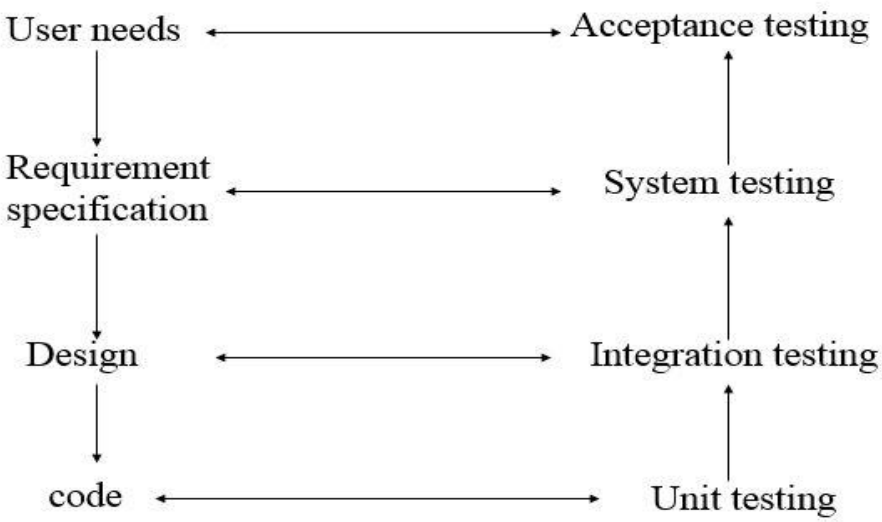


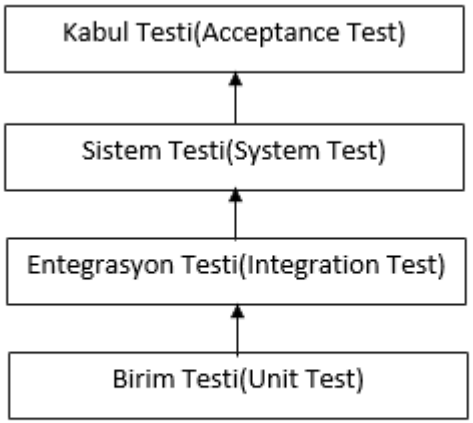
Gri Kutu Testi



Beyaz Kutu Testi

4. Test Seviyeleri

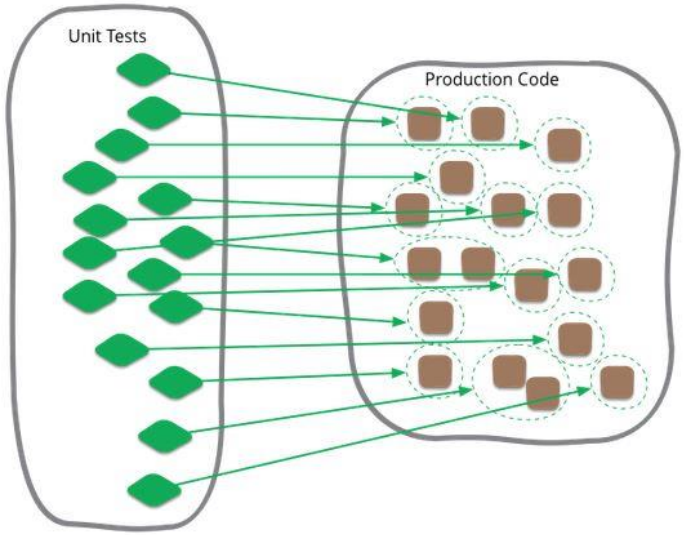




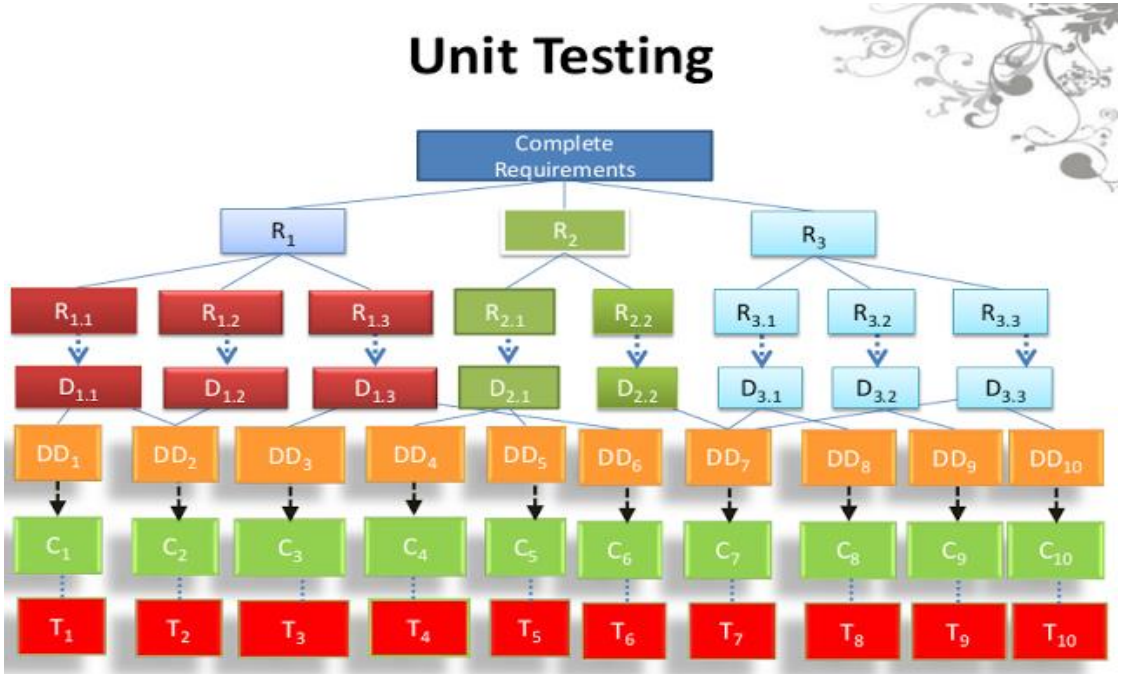
Yazılım Test Seviyeleri(Software Test Levels)

4.1. Birim(Unit) Testi

- Birim testi, yazılım tasarımının en küçük biriminin (yazılım bileşeni yada modül) doğrulanmasıdır. Önemli kontrol yolları, modülün sınırları içerisindeki hataları ortaya çıkarmak için test edilir. Birim testi bir bileşenin sınırları içindeki mantık ve veri yapıları gibi iç işlemler üzerinde çalışır. Bu test türü birden fazla bileşen için paralel olarak gerçekleştirilebilir.



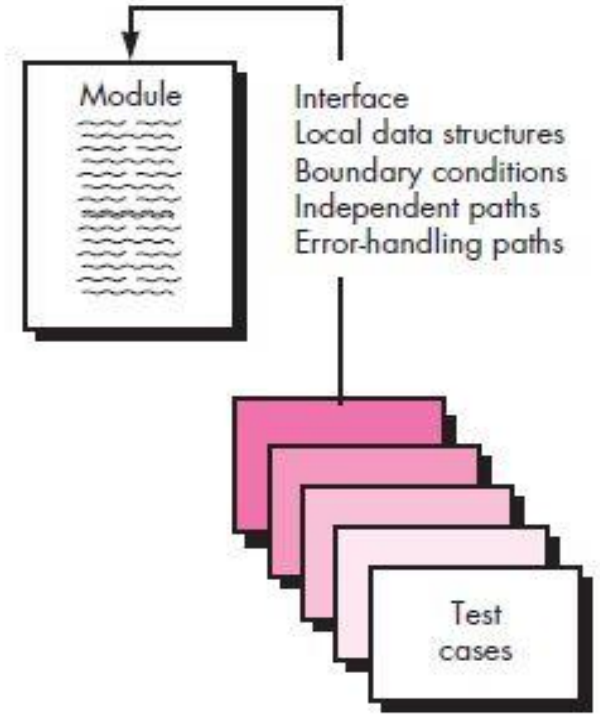
4.1. Birim Testi



4.1. Birim Testi

Birim test durumları

- Ara yüz
- Yerel veri yapıları
- Sınır koşulları
- Bağımsız yollar
- Hata yakalama yolları

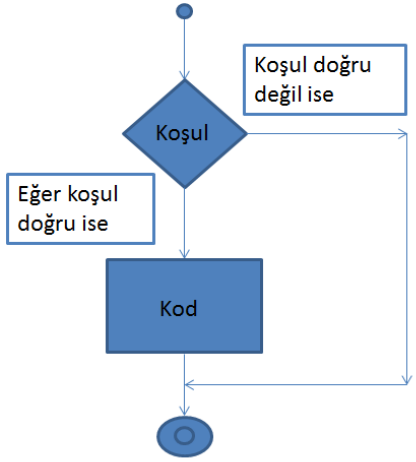


4.1. Birim Testi

- Birim ara yüzü test edilerek bilgi giriş/çıkışlarının uygun ve yeterli şekilde yapıldığı kontrol edilir. Örneğin, programa giren ve çıkan tüm iletilerin doğru tipte oldukları gerçeklenir.
- Yerel veri yapıları incelenerek algoritmanın çalışması boyunca ya da yordamların çağrılması sırasında verilerin saklandığı yerin bütünlüğünün bozulup bozulmadığı test edilmelidir.

4.1. Birim Testi

- Sınır koşullarının en düşük ve en yüksek değerleri, bu değerlerin biraz altı ve biraz üstü kullanılarak sınanmalıdır.
- Birim içindeki birbirinden bağımsız tüm çalışma yolları, tüm dallanmalar tek tek sınanmalıdır.
- Birim içindeki tüm hata yakalayıcılar birer birer denenmelidir.



4.1. Birim Testi

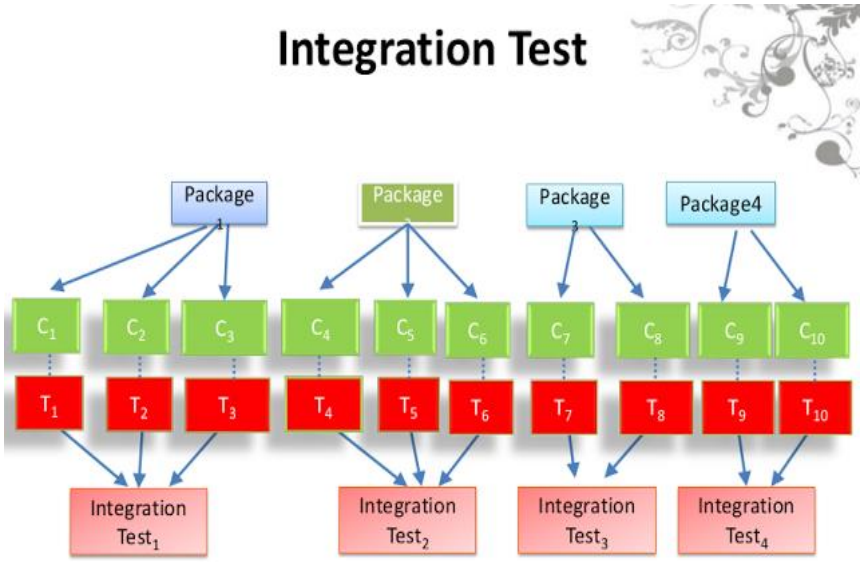
- Daha çok beyaz kutu yönteminin kullanıldığı birim testi ile düzgün ve hatasız çalıştığına kanaat getirilen bir birim artık tümleştirme testi için hazır hale gelmiş olur.

4.1. Birim Testi

- Yazılan kodun her satırının başka bir kod tarafından test edilmesini sağlar.
- Kodun anlaşılmasını kolaylaştırır.
- Daha hızlı yazılım geliştirmeyi sağlar.
- Koddaki hata oranını azaltır.
- Kodların kalitesinin artmasını sağlar.
- Hataların çabuk tespit edilip düzenlenmesini sağlar.

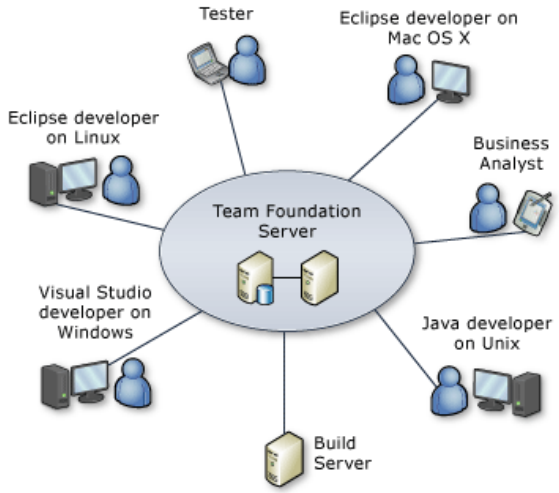
4.2. Tümleştirme(Integration) Testi

- Birden fazla biriminin bir araya getirilerek uyumlu bir şekilde ve hatasız çalışması, her birinin tek tek değil de bir bütün içinde, tasarımda belirtildiği şekilde kendi üzerlerine düşen görevleri yerine getirip getirmediği tümleştirme testi ile kontrol edilir.



4.2. Tümleştirme Testi

- Çoğu zaman, bireysel olarak doğru çalıştığı sanılan yazılım birimleri, bir araya getirildiklerinde daha önceden fark edilemeyen veya öngörülemeyen davranışlarda bulunabilirler. Bu kusurlu davranışları yakalayabilmek için yapılan tümleştirme testi, birimler arasındaki ara yüzlerden kaynaklanan kusurları ortaya çıkarabilmek ve program yapısını oluşturmak için uygulanan sistematik bir tekniktir. Amaç, birim testlerini başarı ile geçmiş modülleri alıp tasarımda belirtilen program yapısını ortaya çıkarmaktır.



4.2. Tümleştirme Testi

- Tümleştirme testinin yapılma nedenleri:

✓Bir birimin çalışması başka bir birimin çalışmasını etkileyebilir.

✓Birimler arasındaki arayüzler arasında verilerin kaybolma olasılığı vardır.

✓Bir birim içinde kabul edilebilir sınırlar içinde olan kesinlik değerleri birden fazla birimin devreye girmesi ile kabul edilemeyecek değerlere ulaşabilir.

✓Birimler arasında eşzamanlılığın sağlanması gerekir.

✓Birimler arasında paylaşılan evrensel veri yapıları sorun çıkarabilir.

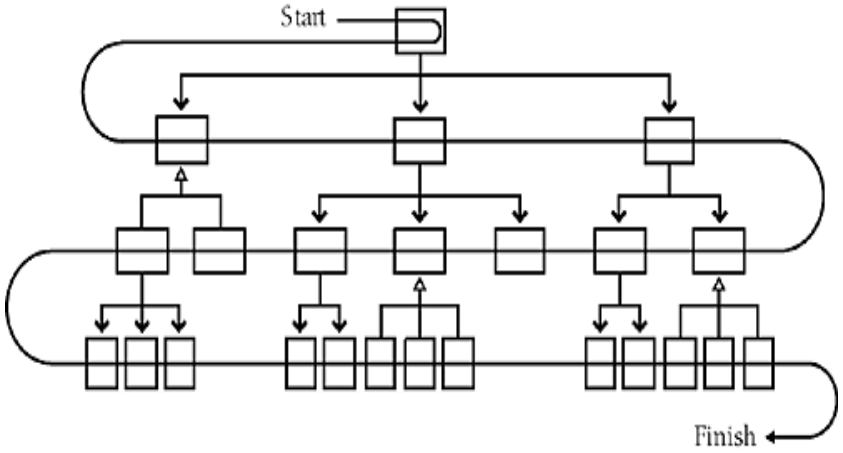
4.2. Tümleştirme Testi

- Birimler bir anda tümleştirmek yerine artırımlı olarak tümleştirmek daha iyi sonuç verir. Artırımlı tümleştirme yönteminde değişik stratejiler kullanılabilir. Bunlar

  - Yukarıdan aşağı tümleştirme
  - Aşağıdan yukarıya tümleştirme
  - Geri çekilme(Regression) testi

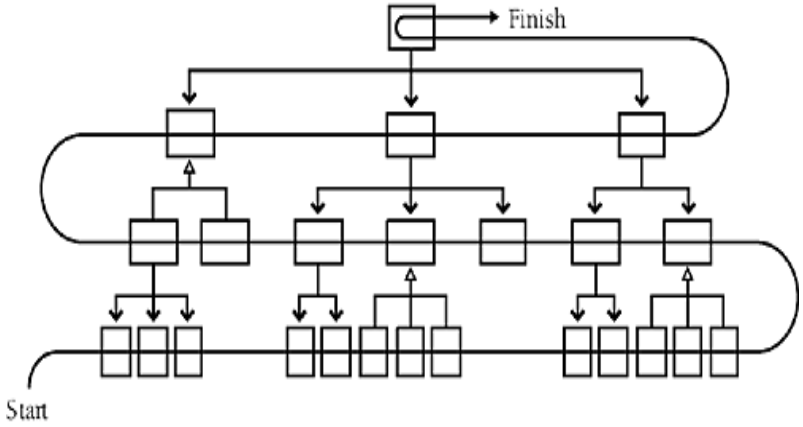
4.2.1. Yukarıdan Aşağı Tümleştirme

- Bu stratejide, önce ana denetim biriminin testi yapılır, sonra ona en yakın düzeydeki birimlerden biri ile beraber test yapılır.



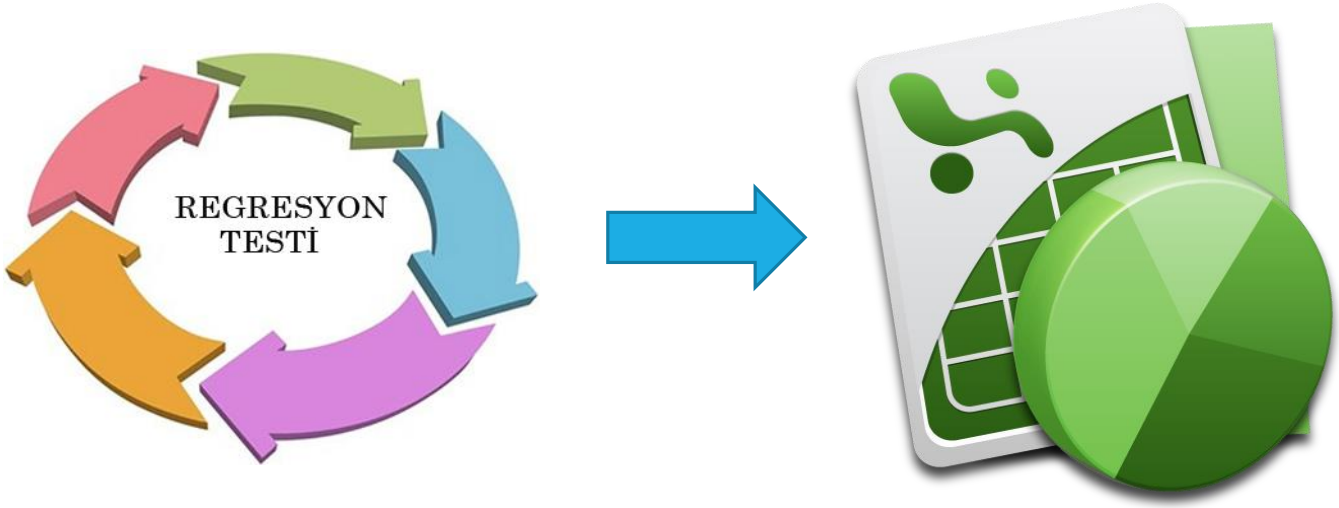
4.2.2. Aşağıdan Yukarıya Tümleştirme

- Alt düzey birimler birleştirilerek kümeler haline getirilir. Bu kümeler test edilir. Daha sonra bu kümelerin birleştirilmesinden oluşan daha üstü düzeyde kümeler meydana getirilir. Bu şekilde en üstte bulunan ana birime kadar ulaşılır.



4.2.3. Regresyon Testi

- Modül eklendiği veya değiştirildiği zaman yazılım değişir. Yeni veri akış yolları oluşur, yeni giriş/çıkışlar meydana gelir ve yeni mantık yapıları çağırılır. Bu değişiklikler daha önce sorunsuz olarak çalışan fonksiyonlarda problemlere sebep olabilir. Tümleştirme test stratejisi kapsamında regresyon testi; uygulama ortamındaki yapılan tüm değişikliklerin yeni bir hata üretip üretmediğini kontrol amaçlı olarak yapılan test türüdür.



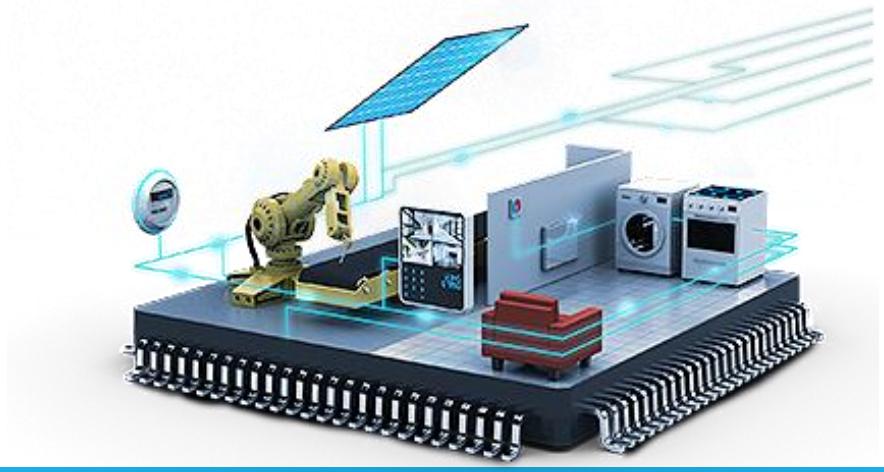
4.3. Sistem Testi

- Aşağıdaki test türlerini içerir.
  - Performans Testi
  - Yük Testi
  - Germe Testi
  - Kurtarma Testi
  - Taşınabilirlik Testi
  - Kullanılabilirlik Testi

4.3.1. Performans(Performance) Testi

- Performans testi, sistemin belirli durumlarda, belirlenen beklentileri verip vermediğini kontrol etmek amacıyla yapılan testtir. Performans testi sistemdeki hataların bulunmasını amaçlamaz ancak sistemdeki darboğazları ortaya çıkarır.

Performans testlerinde amaç sistemin bir açığını bulmak değildir. Asıl amaç sisteme yapılan girdilerden alınan çıktılarla, olması gereken sonuçların uygunluğunu tespit etmektir.



4.3.1. Performans Testleri

- Örneğin; büyük bir veri tabanı yönetim sisteminin başarımı, arama ve sonucu gösterme işlemleri için gereken süredir. Gömülü bir sistemin performansı, insan katkısı olmadan yapmak zorunda olduğu işlemleri başarıyla yapmasıdır.
- Özellikle gerçek zamanlı sistemler için tanımlanmış olan zaman kısıtlarına uymak hem yazılım hem de donanım bileşenleri tarafından karşılanması gereken çok önemli isterlerdir.
- Bu isterlerin uygun şekilde karşılanıp karşılanmadığını görebilmek için tümleştirilen yazılım ve donanım üzerinde performans testleri yapılır. Performans testleri tüm test aşamalarında yapılabilir(birim,stress vs.)

4.3.2. Yük (Load) Testi

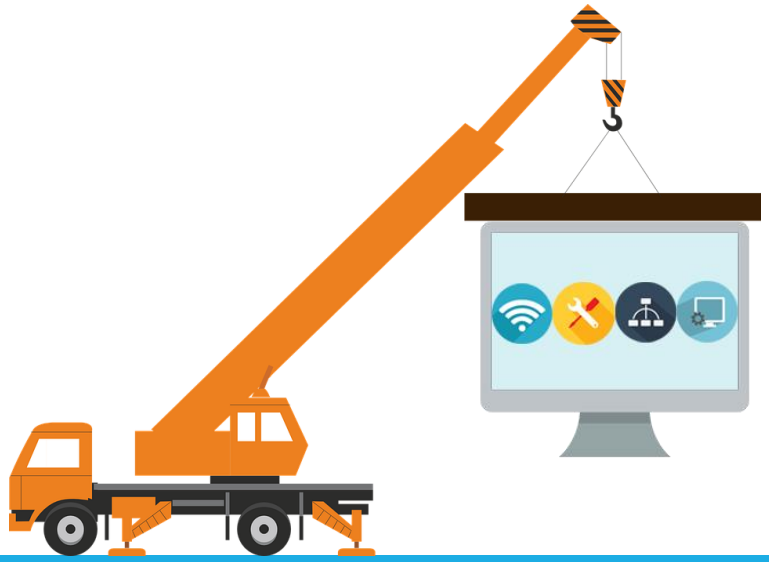
- Her bilgisayar sistemi belirli tür ve miktarda veriyi işlemek ve bunlara göre başka veriler üretmek için tasarlanır. Gerçek zamanlı sistemler ve kontrol sistemleri daha çok belirli kesmelere karşı bir işlem yaparak belirli bir tepkide bulunur. Veritabanı yönetim sistemleri ise çok miktarda veriyi saklama, bunlar üzerinde sorgu yapma ve rapor üretme gibi işlevler yürütür. İşte bu tür yoğun veri akışına sahip sistemler için yükleme testleri yapılır.

4.3.2. Yük Testi

- Yük testleri, sistemin sınırlarını zorlayarak en fazla ne kadar veri işleme kapasitesi olduğunu belirlemek, bu yükte davranışlarını kontrol etmek amacıyla yapılır. Hatta, bazı durumlarda, isterlerde belirtilen değerlerin de üzerine çıkılarak kaldırabilecek en fazla yükün ne olabileceği araştırılır.

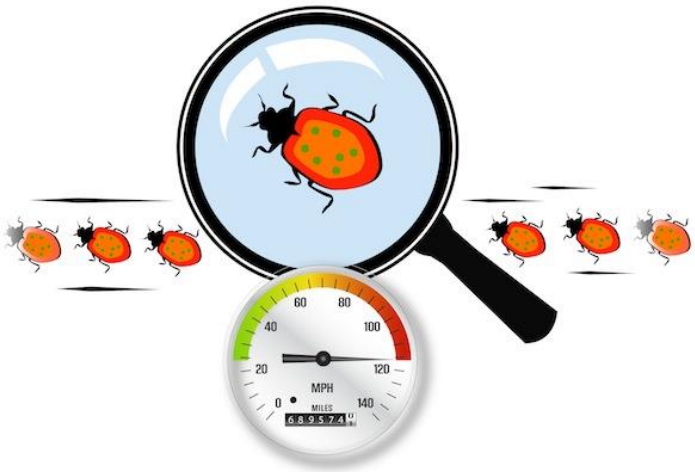
4.3.2. Yükleme Testi

- Tipik bir sistemin yükleme testleri şunlardan oluşabilir:
  - Veri hacmi testi: Sistemin yüksek miktarda veri ile yüklenmesi
  - Veri debisi testi: Tüm girişlerin yüksek hızda veri ile yüklenmesi
  - Kapasite testi: Sistemin bellek ve disk kullanımının zorlanması



4.3.3. Germe(Stress) Testi

- Normal olmayan koşullarda, hem yazılım hem de donanımın ne şekilde davranacağını görmek üzere germe(stress) testleri yapılmalıdır.



4.3.3. Germe Testi

Bu amaçla yapılacak testler şunlardır:

- Sistemi normal olmayan miktarda öz kaynak gerektirecek şekilde zorlamak amacıyla yapılan testler
  - Yüksek hacim ve hızda veri girişi yapmak
  - Beklenenden çok daha yüksek frekansta kesmeler yapmak
  - Beklenen ve kullanılandan çok daha fazla bellek ve işlemci gücü gerektiren durumlar yaratmak
- Sistemin kaldırabileceği yük durumunda, ani etkilere verilecek tepki süresini ölçmek üzere yapılan testler

4.3.4. Kurtarma(Recovery) Testi

- Bilgisayar sistemlerinin çoğunda bir hata durumunda kendini toparlayarak tekrar çalışmaya devam etmesi beklenir. Aşağıdaki yöntemler kullanılarak zararlar en aza indirilebilir.
- Yedekli yazılım mimarisinde, ana yazılım birimi çalışırken, yardımcı yazılım da aynı veya farklı bir donanım üzerinde paralel şekilde çalışır, fakat çıktı üretmez. Ana yazılımın çökmesi halinde bu yardımcı yazılım devreye girer.
- Hataya dayanıklı yazılım ise, herhangi bir nedenle bütünüyle çökmemek üzere, kendi kendini düzeltebilen modüller halinde geliştirilen yazılımlardır.

4.3.4. Kurtarma Testi

- Kurtarma testi önce yazılımı sonra da donanımı çeşitli olası şekillerde bilinçli bir şekilde çökerterek sistemin kendini tekrar toplamasının denenmesi, isterlerin doğrulanması amacıyla yapılır. Bu kapsamda genelde şu testler yapılır:

4.3.4. Kurtarma Testi

- Yedekli yazılım mimarisinde, ana yazılımın devreden çıkartılması ve yardımcı yazılımın otomatik olarak devreye girmesi, bilgi işlemenin kayba uğramadığının kontrolü
- Yeniden yazılım modülü başlatma yönteminde, çken modülün tekrar başlatılması ve çökmeden önceki durumunu kazanması
- Çökme sırasında kaybolma olasılığı olan verilerin tekrar alınması veya üretilmesi
- İnsan katkısı gereken geri kazanma durumlarında, ortalama zamanın ölçülmesi, isterlere göre değerlendirilmesi

4.3.5. Güvenlik Testi

- Güvenlik testi, bir bilgi sisteminin verileri ve işlevselliğini korumak için tasarlanmış bir süreçtir. Herhangi bir bilgi sızıntısı olup olmadığını kontrol edilir. Sistemin tüm potansiyel açık kapıları ve zayıflıkları araştırılır.

4.3.5. Güvenlik Testi

- Temel güvenlik testi çeşitleri şunlardır:
  - Zafiyet taraması
  - Penetrasyon Testi
  - Risk Belirleme
  - Güvenlik Denetimi
  - Şifre Kırma

4.3.6. Taşınabilirlik (Portability) Testi

- Taşınabilirlik testi, var olan bir yazılım bileşeni veya uygulamayı yeni bir ortamda test etme işlemidir. Uygulamanın diğer ortamlarda
  - Installability
  - Combatibility
  - Adaptability
  - Replaceability

yetenekleri araştırılır. Sonuçlar rapor edilir.

4.3.6. Taşınabilirlik Testi

- Uygulamalar şu ortamlar için test edilebilir:
  - Donanım platformları(istemciler, sunucular, ağ bağlantı cihazları, giriş ve çıkış aygıtları)
  - İşletim sistemleri (versiyonları ve servis paketleri dahil).
  - Tarayıcılar (her bir sürümü dahil)

4.3.7. Kullanılabilirlik(Usability) Testi

- Tasarımların veya ara yüzlerin kullanıcı ile buluşmasından önce tasarımın kullanılabilirliğini ölçmek amacıyla yapılan testlere denir. Sadece bir kullanıcının o an için hareketlerini gözlemlemek ile ilgilidir. Kullanılabilirlik testleri kullanılabilirlik problemleri hakkında bize bilgi verir ve kullanıcıların uygulama ile nasıl etkileşimde bulunduğuna bakar.

4.3.7. Kullanılabilirlik Testi

- Kullanılabilirlik, 5 niteliksel özellik ile ölçülür.

Öğrenilebilirlilik: Kullanıcılar, tasarımı ilk kullandıklarında yerine getirmeleri gereken görevleri kolaylıkla yapabiliyorlar mı?
Verimlilik: Kullanıcılar, tasarımın çalışma şeklini öğrendikten sonra gerçekleştirecekleri işlemleri ne kadar hızlı yapabiliyorlar?
Memnuniyet: Tasarımı kullanmak kullanıcıları duygusal anlamda mutlu ediyor mu, kullanıcılar tasarımı kullanırken kendini rahat hissediyorlar mı?

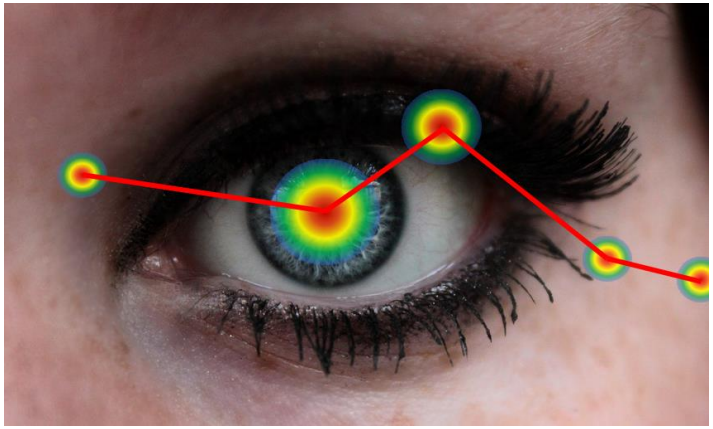
4.3.7. Kullanılabilirlik Testi

Hatırlanabilirlilik: Kullanıcılar, bir süre tasarımı kullanmadıktan sonra tekrar kullanamaya başladıklarında tasarıma dair var olan bilgilerin ne kadarını hatırlayabiliyorlar?

Hatalar: Kullanıcılar, ne kadar hata yapıyor ve bu hataları ne sıklıkta tekrarlıyorlar, hataları ne kadar hızlı yok edebiliyorlar?

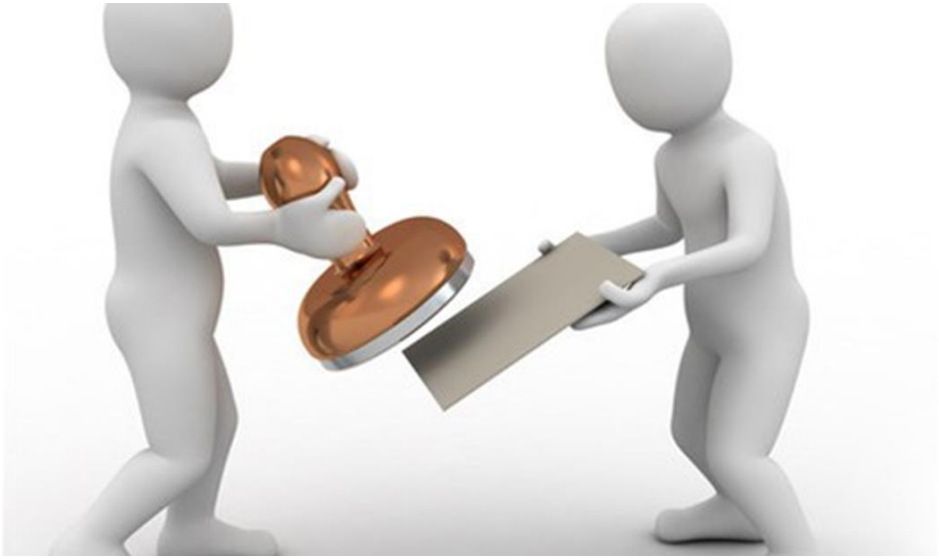
4.3.7. Kullanılabilirlik Testi

- Örnek olarak, bir web sitesinin kullanılabilirlik testini ele alalım. Göz izleme cihazı, kullanıcın nereye, ne kadar süre ve kaç kere baktığına, anlık ve geçmiş dikkatinin nerede yoğunlaştığına, niyetine ilişkin bilgi sağlar. Göz izleme cihazı göz bebeklerinin hareketlerini ve odaklanmalarını izleyerek odak noktaları haritası çıkarır. Bu bilgilere göre web sitemizi kullanıcının ilgisini çekebilecek ve daha kolay kullanımını sağlayacak şekilde düzenleyebiliriz.



4.4. Kabul Testi

- Çalıştırılmadan önce yazılımın son sınanmasıdır. Artık yapay veriler yerine gerçek veriler kullanılır. Bu sınama türü;
  - Alfa sınaması
  - Beta sınaması olarak çeşitlenir.



4.4.2. Alfa Testi

- Geliştiricinin kendi yerinde müşteri tarafından yapılır. Geliştirici bu testleri gözlemleyerek gerçek kullanım hakkında bilgi sahibi olmaya çalışır; kusur buldukça not alır ve düzenleme işlemlerini yürütür. Alfa testlerinin en önemli özelliği, denetim altındaki bir ortamda, asıl kullanıcılardan biri tarafından yapılıyor olmasıdır.

4.4.2. Beta Testi

- Birçok kullanıcının kendi ortamında yapılır. Geliştirici genellikle bu testlere katılmaz; yalnızca belirli aralıklarla sonuçları ve yorumları alır. Bu testin özelliği de geliştirici tarafından kontrol edilemeyen gerçek uygulama ortamı koşullarında yazılımın denenmesidir. Beta testi sonunda geliştirici, bulunan kusurları düzelterek tüm kullanıcılar için yeni bir sürüm çıkartır.

Çalışma Soruları

1. Yazılım test seviyelerini şekil çizerek gösteriniz.

2. Otomatik test ile manuel test arasındaki farkları yazınız.

3. Birim testi nedir? Birim testi ile bir uygulamada neler test edilir?

4. Tümleştirme testi nedir? Çeşitlerini yazınız.

5. Taşınabilirlik testi ile uygulamalar hangi ortamlar için test edilirler? Açıklayınız.

6. Kullanıcı kabul testleri yazıp açıklayınız.

7. Kara, beyaz ve gri kutu tekniklerini açıklayıp birer örnek veriniz.

8. Kurtarma testi nedir? Hangi durumlarda gereklidir?

9. Güvenlik testi neden gereklidir? Güvenlik testi ile bir uygulamada neleri test ederiz?

Kaynaklar

[1] Software Enginnering A Practitioner's Approach (7th Edition),Roger Pressman,2013

[2] Yazılım Mühendisliği(2. Baskı), M.Erhan Sarıdoğan,2008

[3] http://www.tutorialspoint.com/software_testing/software_testing_quick_guide.htm

[4] http://www.testrisk.com/2012/08/test-sozlugu.html

[5] https://en.wikipedia.org/wiki/Security_testing

[6] http://www.slideshare.net/pbaskarmca/security-testing-4338585

[7] http://blog.milliyet.com.tr/yazilim-kullanilabilirlik-testleri/Blog/?BlogNo=463001