YMT 412-Yazılım Kalite Ve Güvencesi

Test Süreci Ve Yönetimi

Bölüm-7

Fırat Üniversitesi Yazılım Mühendisliği Bölümü

İçindekiler

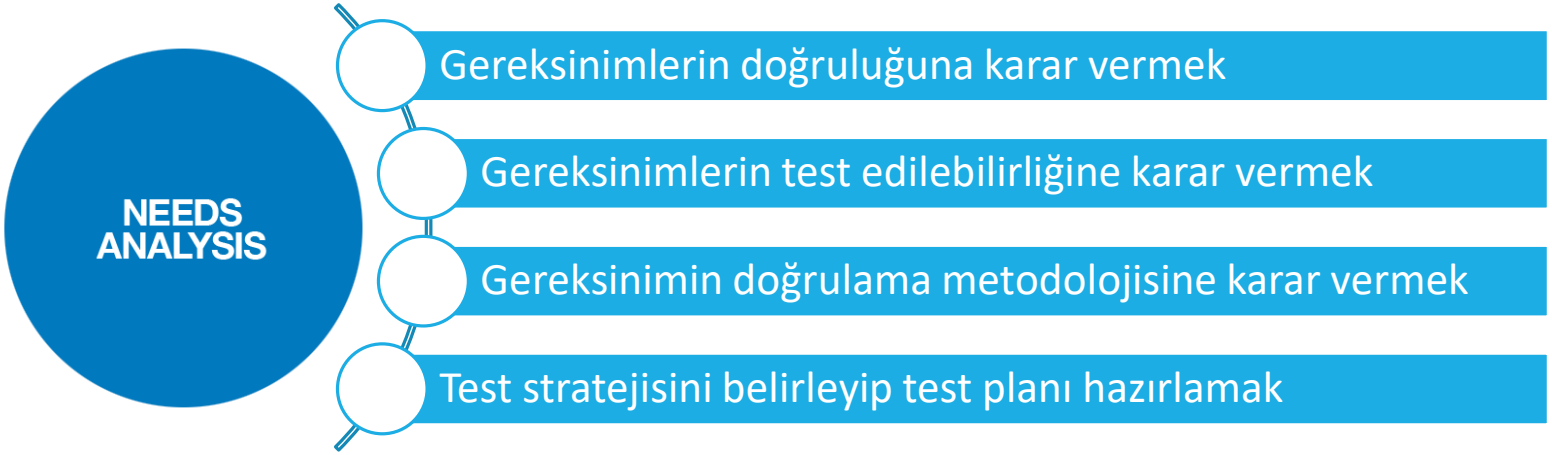
1. Test Süreci Ve Yönetimi

- Bir yazılım geliştirirken, yazılımın en az derecede hata içermesi ve kendinden beklenenleri en üst seviyede karşılaması için yazılım test eylemleri, yazılım geliştirme sürecinde en erken safhada başlamalıdır. Testçiler, erken safhalarda başlayan test eylemleri ile muhtemel hataları yazılım geliştirme sürecinin en erken safhalarından itibaren bulmayı ve bulunan hataların düzeltilmesini amaçlar.

1. Test Süreci Ve Yönetimi

- Test eylemleri yazılım yaşam döngüsünün şu aşamalarında gerçekleştirilir.

- Gereksinim analizi:



1. Test Süreci Ve Yönetimi

- Tasarım
  - Kabul, sistem ve tümleştirme test prosedürlerini ve test durumlarını üretmek
  - Birim test stratejisini belirlemek
  - Testler için gerekli test verilerini üretmek ve doğrulamak
  - Hata bildirim yapısını tanımlamak ve işlerliğini denemek
  - Testler için gerekli test ortamını tanımlamak

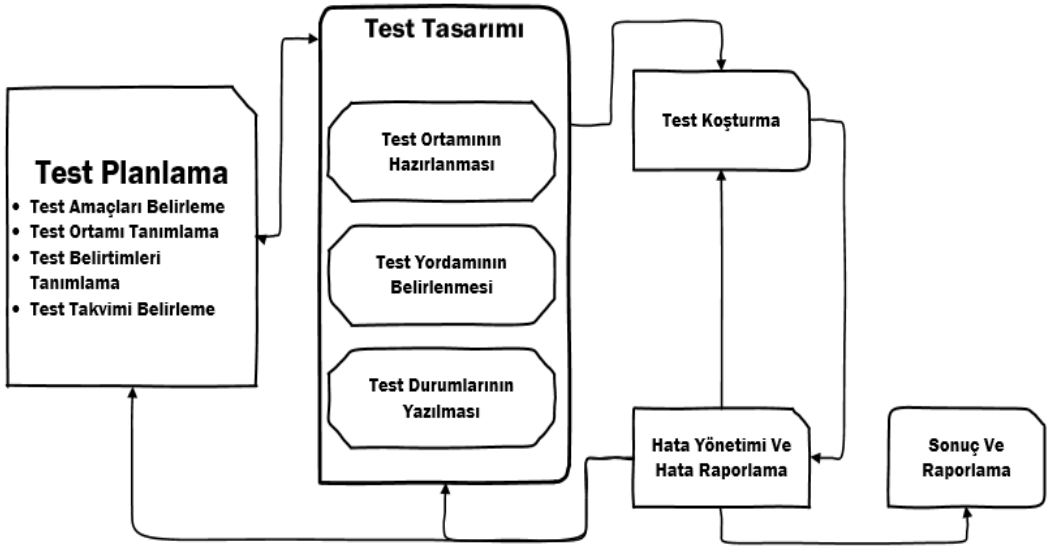
1. Test Süreci Ve Yönetimi

- Kodlama
  - Birim testlerin gerçekleştirilmesini izlemek ve sonuçlarını denetlemek
  - Kod gözden geçirmelerini izlemek ve sonuçları denetlemek
  - Bütünleştirmek ve sistem testleri için test ortamını hazırlamak

1. Test Süreci Ve Yönetimi

- Test
  - Tümleştirme, sistem ve kabul testlerini gerçekleştirmek
  - Bulunan hataları bildirmek ve düzeltildiğini denetlemek
  - Yineleme testlerini gerçekleştirmek
- Bakım
  - Yineleme testlerini gerçekleştirmek

2. Yazılım Test Süreci



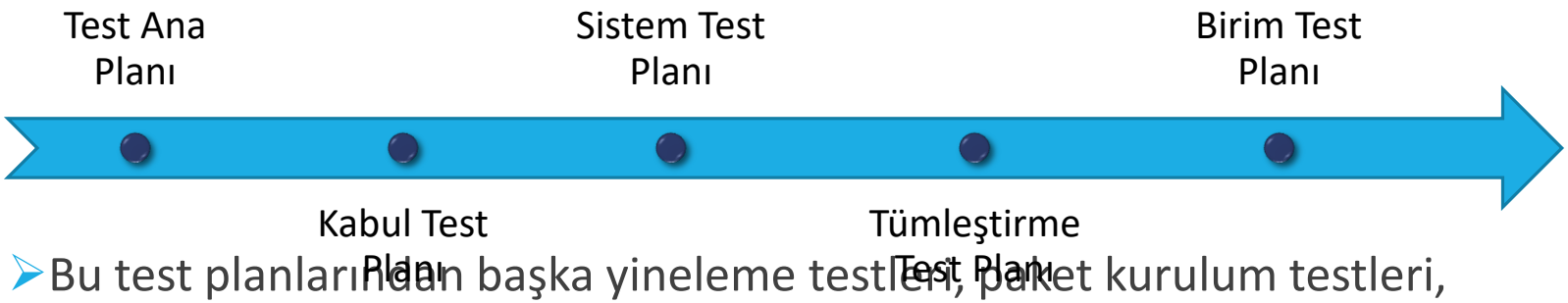
Yazılım test süreci önce planlanan, sonra icra edilip sonuçları kayıt altına alınarak belgelendirilen bir dizi eylemden oluşur. Yandaki şekilde yazılım test süreci görülmektedir. Bu süreç geliştirilen yazılımdaki hataların varlığına odaklanır.

2.1. Test Planlama

- Yapılacak testler projenin başında planlanmalı ve belgelenmelidir. Bir test planı testin kapsamını, testin stratejisini, test ortamını, hangi yazılım parçalarının test edilip edilmeyeceğini, proje kapsamında amaçlanan test eylemlerini, kaynakları ve takvimi içeren bir dokümandır.
- Test planı geliştirilirken basit, tam, anlaşılır, güncel ve içerdiği eylemler bakımından kabul edilebilir olmalıdır.
- Yazılım projelerinde test planlamasında iki farklı yaklaşım takip edilebilir:

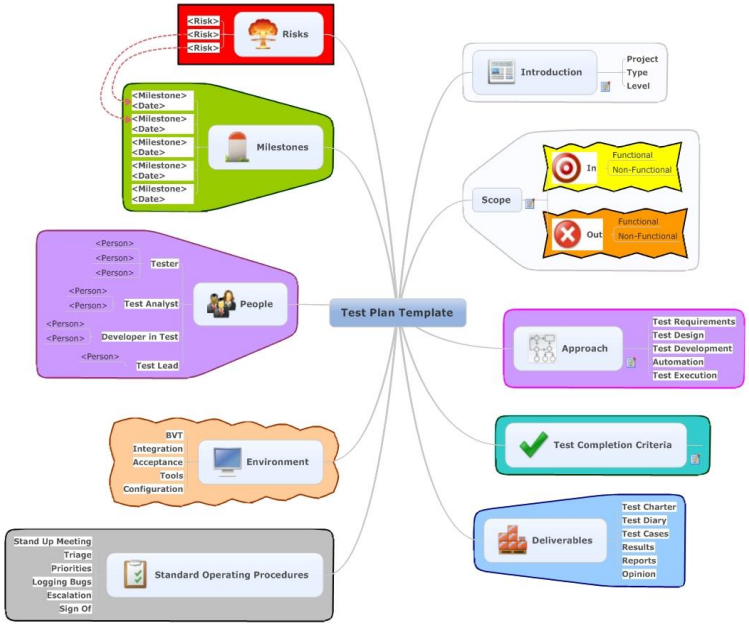
2.1. Test Planlama

- Birinci Yaklaşım: Bu yaklaşımda test planlama stratejisi, her bir seviye test için ayrı test planı geliştirilmesine dayanır. Projenin genel test stratejisi Test Ana Planı adı verilen bir plan içerisinde belirtilir. Bu yaklaşıma göre geliştirilebilecek test planları şunlardır:



- Bu test planlarından başka yineleme testleri, paket kurulum testleri, donanım testleri ile ilgili planlar da geliştirilebilir.

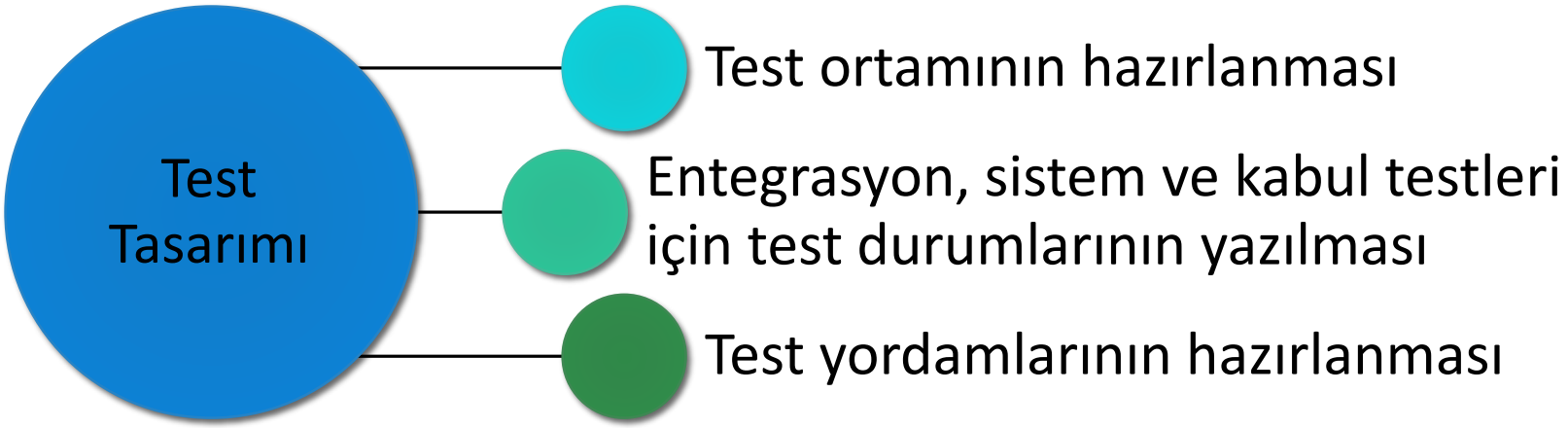
2.1. Test Planlama



- İkinci Yaklaşım: İcra edilecek tüm testler için tek bir test planı geliştirilir. Genel olarak Kabul Test Planı veya Sistem Test Planı diye adlandırılır. Geliştirilen plan, birim, tümleştirme, sistem ve kabul testlerinin ve diğer testlerin planlamalarını da kapsar.

2.2. Test Tasarımı

- Test planlama süreci tamamlandıktan sonra test tasarım süreci başlar. Bu süreçte aşağıdaki görevler icra edilir:



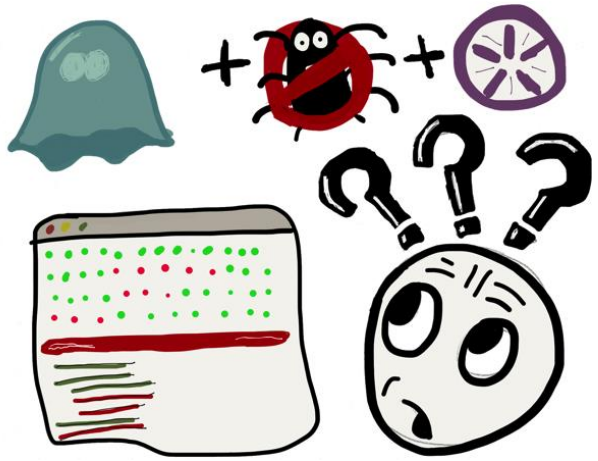
2.2.1. Test Ortamının Hazırlanması

- Testler test mühendisleri tarafından tanımlanan yazılım ve donanımdan oluşan bir ortamda gerçekleştirilir. Test ortamı hazırlanırken testlerde kullanılacak olan yardımcı test yazılımları da geliştirilir veya hazır alınır. Yazılım testlerinde kullanılan yardımcı yazılımlar şunlardır:

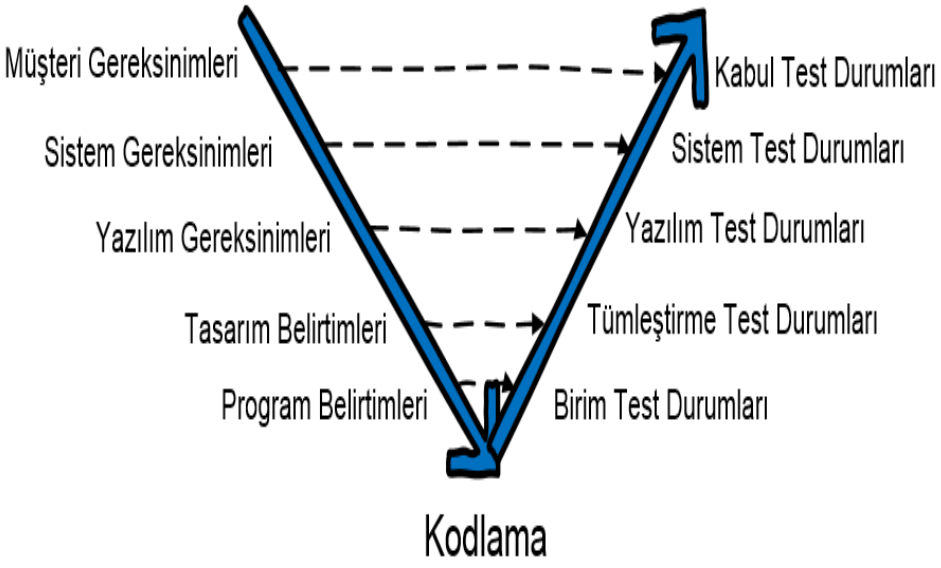
- Koçan (Stub) ve Sürücüler: Testler için gerekli olan sistemin işlevsel olmayan bileşenlerinin yerini tutan ufak yazılım parçacıklarıdır.

2.2.1. Test Ortamının Hazırlanması

- Emülatör ve Simülatörler: Yazılımların ihtiyaç duyduğu gerçek donanımları taklit eden ve testler için gerekli olan donanım verilerini sağlayan yazılımlardır.
- Test Verisi Üreteçleri: Test durumlarının koşturulması için gerekli olan test girdi verilerini üreten yazılımlardır.
- Hata Ayıklayıcılar(Debugger): Testler sırasında karşılaşılan hataların kaynak kod üzerinde bulunmasını sağlayan yazılımlardır.



2.2.2. Test Durumlarının Yazılması

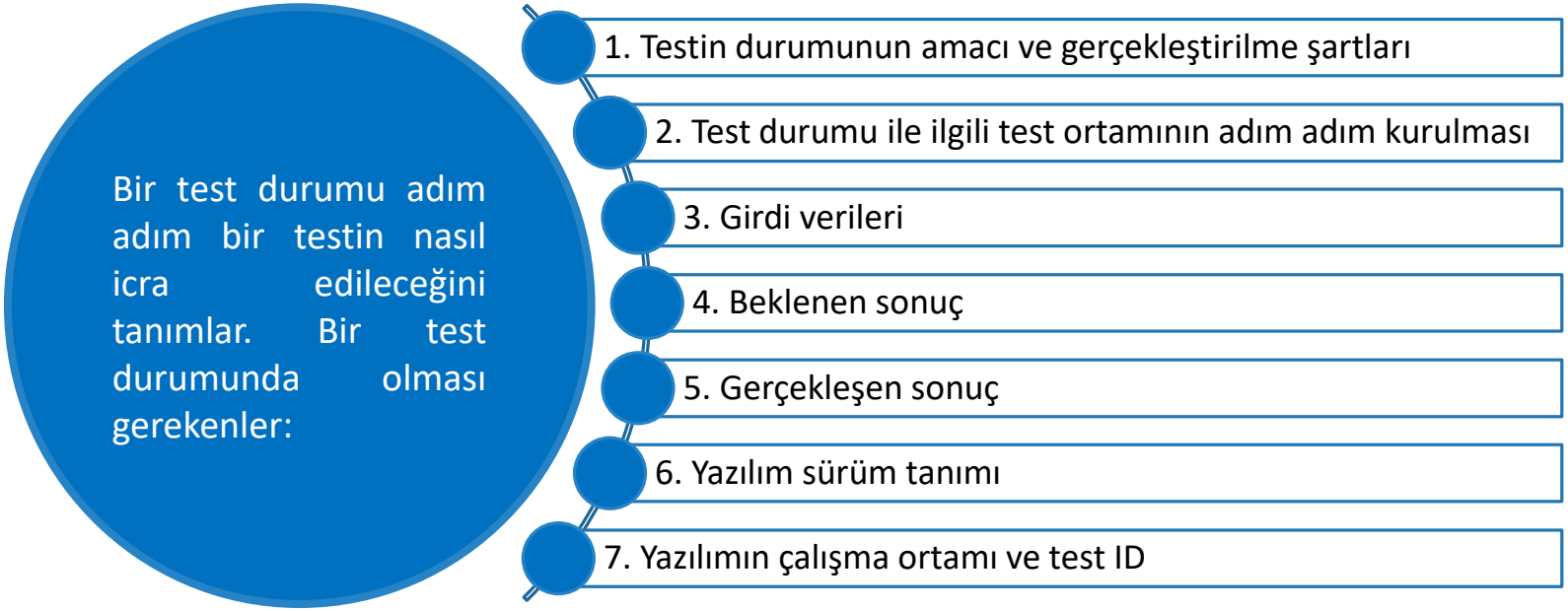


Test durumu, belirli bir program parçasının çalıştığının veya bir gereksinimin doğrulandığının gösterilmesi için kullanılan girdiler, gerçekleştirilmesi gereken adımlar ve beklenilen tüm sonuçların belirtilmesidir. Test durumları testin en küçük parçasıdır.

2.2.2. Test Durumlarının Yazılması

- Test ekibi resmi hale gelen gereksinimlerden test durumu oluşturmaya başlar. Bir gereksinim, bir test durumu ile doğrulanabildiği gibi birden fazla test durumu ile de doğrulanabilir. Bu amaçla, geliştirilen yazılımın kendi belirtimlerinin tümünü karşıladığını göstermek için her bir gereksinime en az bir tane test durumu yazılmalıdır. Geliştirilen yazılımın kendinden beklenen tüm davranışları gerçekleştirdiğinin göstergesi, yazılıma ait tüm test durumlarının testlerden geçmesidir.

2.2.2. Test Durumlarının Yazılması

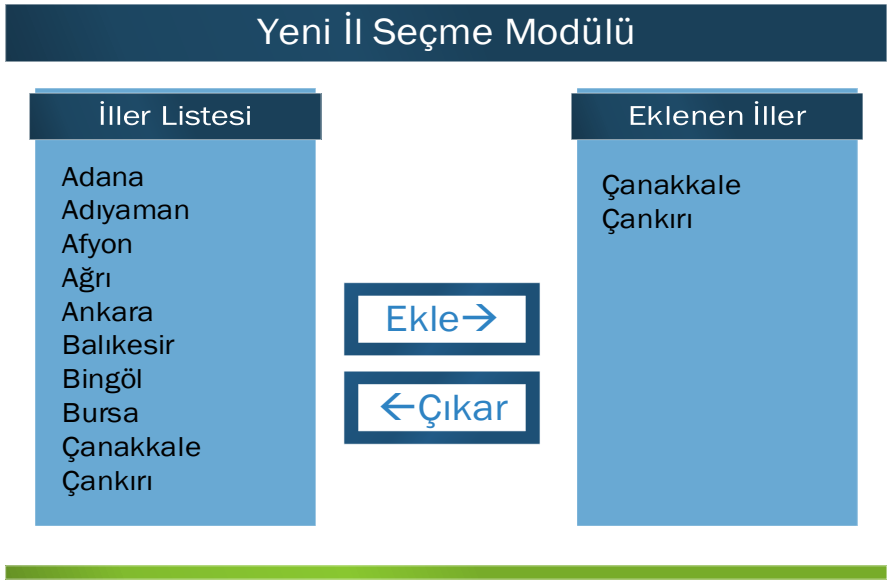


2.2.2. Test Durumlarının Yazılması

- Test durumlarında bilinen girdilere yer verilmelidir. Bu girdiler ile beklenen çıktılar test durumları içerisinde doğrulama noktası olarak verilir. Eğer beklenen girdiler ile beklenen sonuç yazılım tarafından verildi ise ilgili test durumu geçmiştir. Test edilen öğeye ait test durumlarının tamamının veya belirlenen bir oranının testten geçmesi ile gerçekleştirilen testler başarılı sayılabilir. Ancak bu karar projenin test stratejisi kapsamında belirlenmelidir.

- Bir sonraki sayfada verilen örnekte bir projenin yazılım gereksinim belgesinden örnek bir bölüm verilmiştir.

2.2.2. Test Durumlarının Yazılması



Örnek yazılım gereksinimleri

1.1. İl Ekleme

Etiket: «Ekle→» şeklindedir.

İşlevi:

1. Bu tuşa basıldığında İller Listesinde seçili olan il Eklenen İller Listesine eklenir.

2. İller Listesinde bir il seçili değilse «Öncelikle İller Listesinden Bir İl Seçmelisiniz!» mesajı gösterilir.

2.2.2. Test Durumlarının Yazılması

Test Durumu Formu		
TD NO	**Test Durum Adı**	**İlgili İsterler**
1	İl Ekle Butonu Etiket	1.1
Ön Koşullar		
Program çalışıyor ve Yeni İl Seçme modülü açık		
Test Adımları		
1. İl ekle butonunun etiketinin «Ekle→» şeklinde olduğunu doğrula.		
Test Sonuçları		
Beklenen Sonuç	**Gerçekleşen Sonuç**	**Sonuç(Geçti/Kaldı)**
Resimde görüldüğü gibi		

Örnek test durumları

2.2.2. Test Durumlarının Yazılması

Test Durumu Formu		
TD NO	**Test Durum Adı**	**İlgili İsterler**
2	İl Ekle Butonu İşlev-1	1.1
Ön Koşullar		
Program çalışıyor ve Yeni İl Seçme modülü açık		
Test Adımları		
1. İller listesinden hiçbir il seçme. 2. Ekle→ butonuna bas. 3. «Öncelikle iller listesinden bir il seçmelisiniz» mesajının gösterildiğini doğrula.		
Test Sonuçları		
Beklenen Sonuç	**Gerçekleşen Sonuç**	**Sonuç(Geçti/Kaldı)**
Uyarı mesajı vermesi		

Örnek test durumları

2.2.2. Test Durumlarının Yazılması

- Örnek olarak verilen bu test durumlarının koşturulması ve sonuçlarının kaydedilmesi elle yapılabileceği gibi test betikleri yazarak otomatik olarak da gerçekleştirilebilir.

- Kayıt altına alınan test durumları üç bölümden oluşur. Her bölüm kendi içerisinde aşağıda verildiği gibi bölümlere ayrılır:

1. Giriş/Genel Bakış
2. Test Durumu Eylemleri
3. Sonuçlar

2.2.2. Test Durumlarının Yazılması

1. Giriş/Genel Bakış: Test durumu hakkında genel bilgiler içerir.

Tanımlayıcı: Her bir test durumuna ait olan eşsiz/benzersiz bir tanımlayıcıya sahiptir. Bu tanımlayıcı ile test durumu ilgili test öğesi, test sonucu ve açılan hata bildirimleri ile ilişkilendirilir.

Test durumunu yazan: Test durumunun kimin tarafından yazıldığı bilgisidir.

Sürüm: Test durumunun sürüm numarasıdır.

Adı: Test durumunun ne ile ilgili olduğunu kolayca gösterecek olan bir ifadedir.

Gereksinim Tanımlayıcısı: Test durumunun hangi gereksinim için yazıldığının anlaşılması için gereksinime ait eşsiz tanımlayıcının test durumu içerisinde verilmesi gereklidir.

Amaç: Bu test durumu ile hangi işlevin test edildiğinin belirtilmesidir.

Bağlılıklar: Bu test durumunun gerçekleştirilmesi için varsa kendinden önce gerçekleştirilecek olan test durumlarının eşsiz tanımlayıcılarının belirtilmesidir.

2.2.2. Test Durumlarının Yazılması

2. Test Durumu Eylemleri

Test Ortamı: Test durumunun koşturulabilmesi için gerekli olan yazılım, donanım ve çevresel koşulların belirlenmesidir.

İlklendirme: Test durumu koşturulmadan önce varsa gerekli olan değişkenlerin başlangıç değerlerinin belirtilmesidir.

Sonlandırma: Test durumları koşturulduktan sonra gerçekleştirilecek olan eylemler belirtilir. Örneğin, eğer test durumu veri tabanını siliyorsa, veri tabanının yeniden kayıtla doldurulması gibi.

Eylemler: Testin tamamlanması için yapılacakların adım adım belirtilmesidir.

2.2.2. Test Durumlarının Yazılması

3. Sonuçlar

Beklenen Sonuç: Test durumunda belirtilen tüm adımların gerçekleştirilmesinden sonra test mühendisinin hangi sonuç ile karşılaşacağının belirtilmesidir.

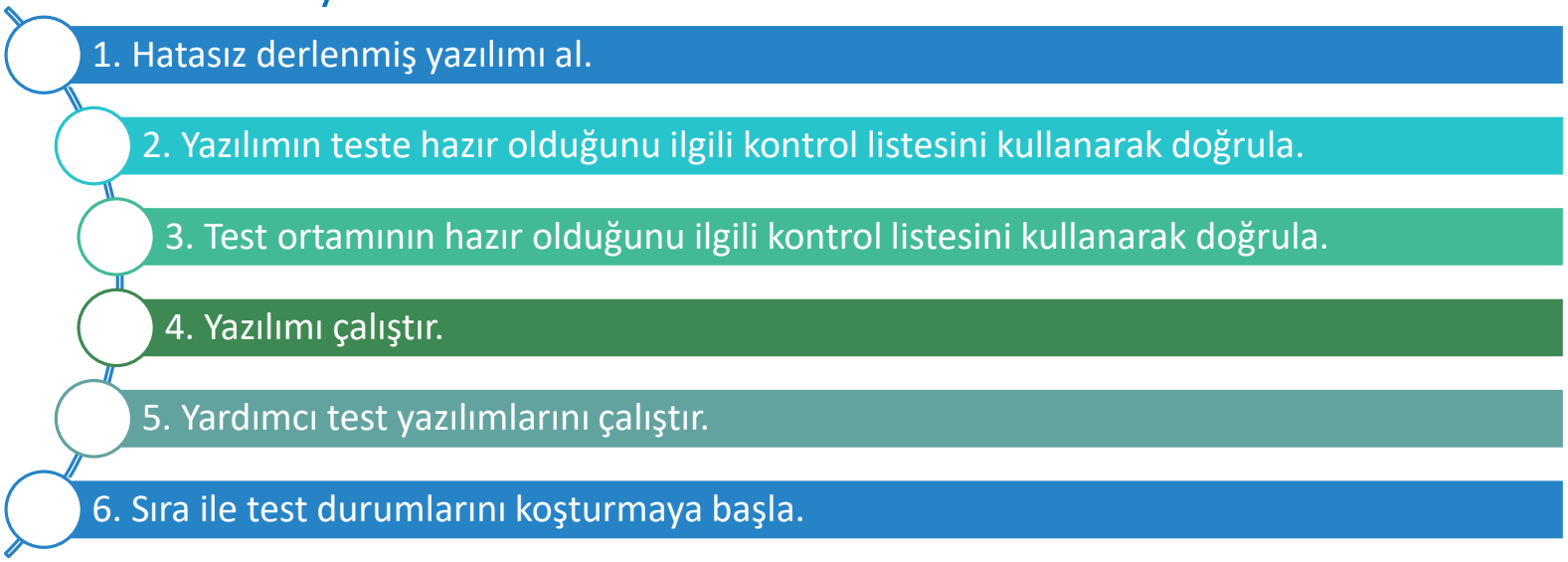
Gerçek Sonuç: Test gerçekleştirildikten sonra ortaya çıkan sonucun belirtildiği alandır. Genellikle bu sonuç ile beklenen sonuç karşılaştırılarak testin geçip kaldığı belirlenir.

2.2.3. Test Yordamı

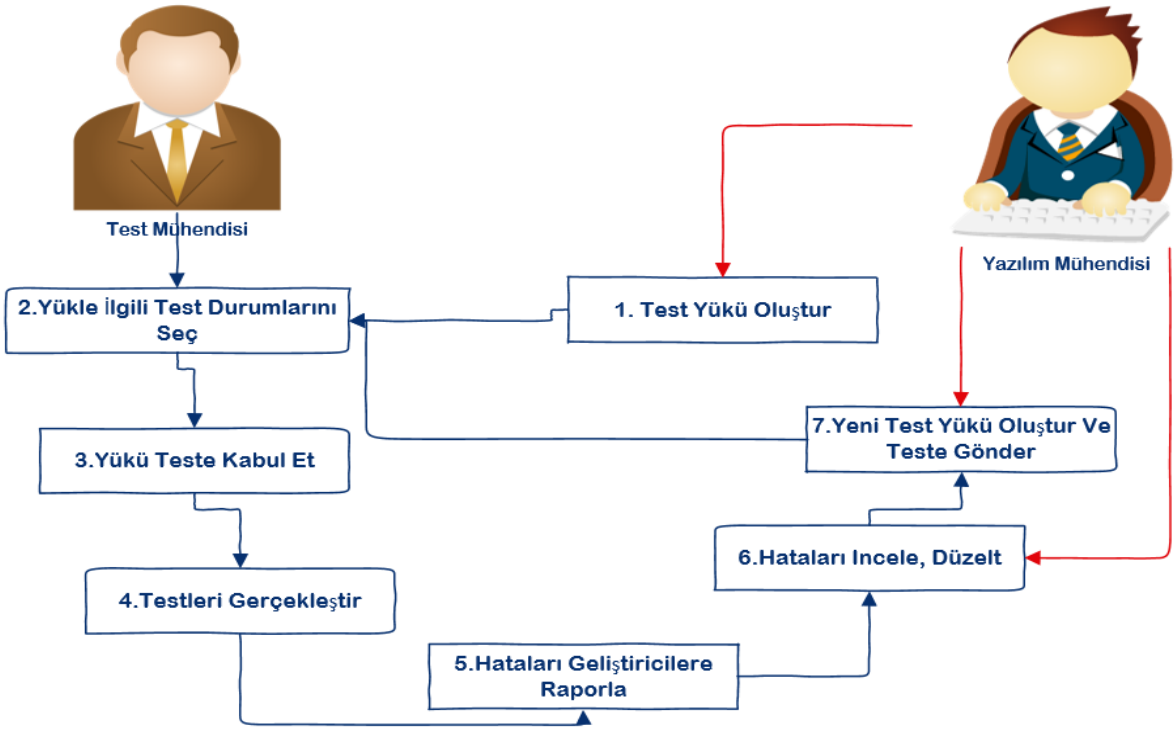
- Test yordamı, her bir test durumunun test ortamının kurulması, koşturulması ve sonuçlarının değerlendirilmesi için ayrıntılı direktifler, açıklamalar listesi içeren ve test planı temel alınarak geliştirilen belgedir.
- Bu belge içerisinde testin gerçekleştirilmesi için adım adım tanımlanmış ayrıntılı açıklamalar vardır.

2.2.3. Test Yordamı

Örnek bir test yordamı:



2.3. Test Koşturma/Gerçekleştirme



2.3. Test Koşturma/Gerçekleştirme

- Test koşturmada genel olarak şu adımlar izlenir.

1. Yazılım ekibi tarafından test edilecek yazılım(yük) oluşturulur.
2. Testçiler gelen yazılıma uygulanacak olan test durumlarır
3. Yazılımın test edilebilir olduğuna karar verilir.
4. Yazılım teste kabul edilirse test başlar.
5. Testlerde bulunan hatalar raporlanarak yazılım ekibine b
6. Bulunan hatalar yazılım ekibi tarafından düzeltilir ve yeni sürüm hazırlanır.
7. Testçiler yeni gelen yük ile yineleme testlerini gerçekleştirir.
8. Bu adımlar müşteriye son kabul edilebilir yazılım verilinceye kadar devam eder.

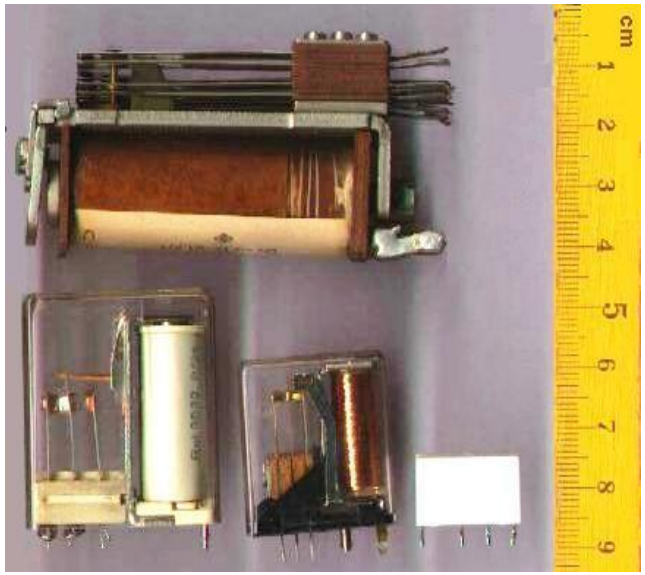
2.4. Hata Yönetimi

- Kullanıcının geliştirilen yazılım ile yapmak istediklerinin yazılım tarafından yapılmaması veya eksik yapılmasına hata denilir. Hatalar genellikle testler sırasında tespit edilir. Testler sırasında belirlenen yazılım hatalarının düzeltilmesi işgücü kaybına ve maliyete neden olur. Testlerden önce, gözden geçirme faaliyetleri sırasında hatalar bulunduğunda daha az maliyetle düzeltilebilir. Proje içinde hataların ve düzeltici faaliyetlerin kontrol altına alınması için bir yaşam döngüsü mevcuttur.

Hata yönetimi yaşam döngüsü

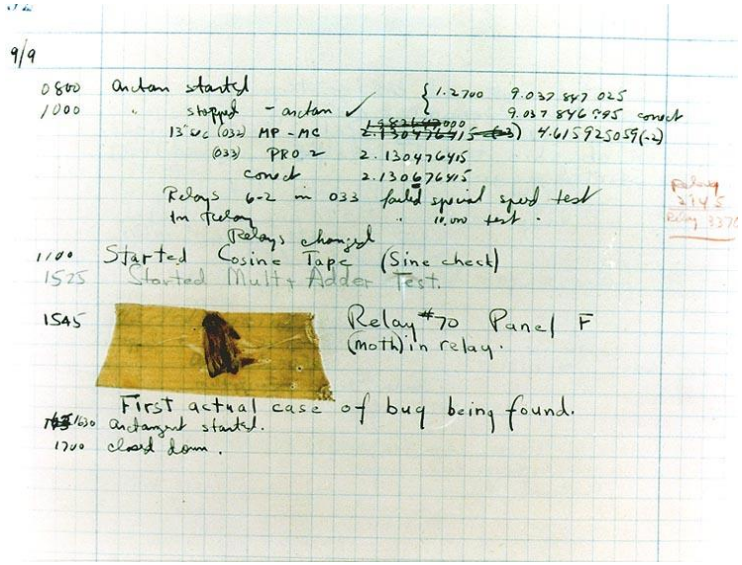
2.4. Hata Yönetimi

- İlk program bug'ı 1947 yılında Grace Murray Hopper'in Harvard Üniversitesi'nde kullandığı Mark II 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.



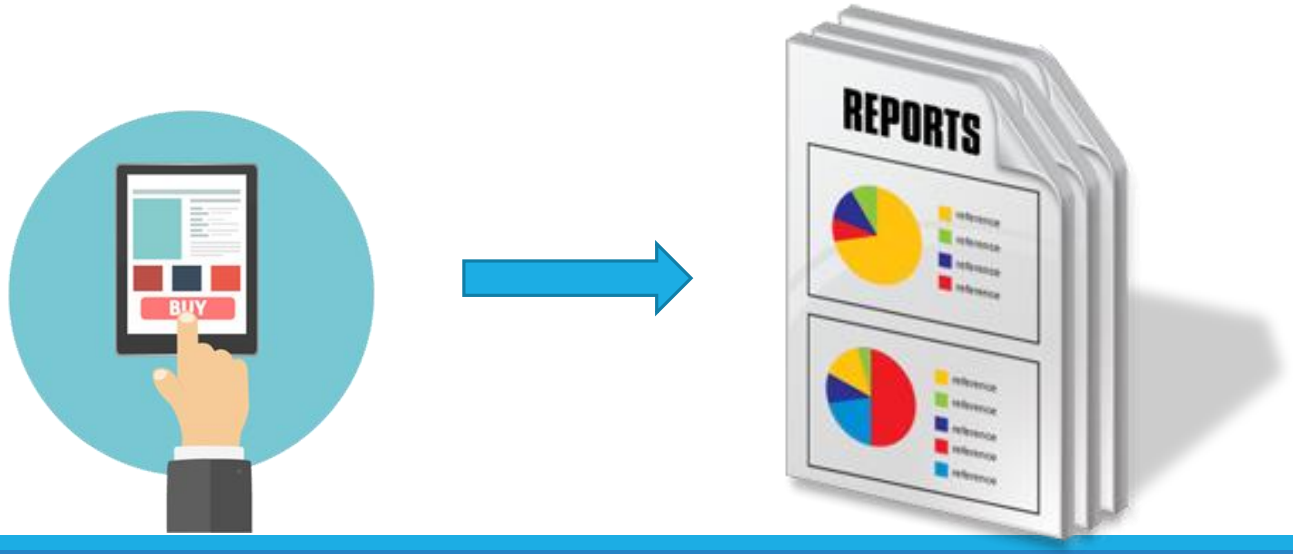
2.4. Hata Yönetimi

- Bilgisayar programlama tarihine ilk program hatası olarak geçen bu böcek operatör tarafından log defterine yandaki şekilde eklendi. Bu işlemin ardından operatörler «hata giderildi» anlamına gelen «debugged (böcek temizlendi)» kelimesini kullanmaya başladılar. Bizim bugünlerde sıkça kullandığımız debug kelimesinin kökeni bu güveden geliyor.



2.4.1. Hata Raporlama

- Yazılım geliştirildikçe hatalar ortaya çıkacaktır. Bu hataların yazılım geliştiricilere bildirilmesi, bildirilen hataların düzeltildikten sonra hatanın kapatıldığının izlenmesi testler açısından önemlidir. En üst düzeyde hatalardan arındırılmış bir yazılımın geliştirilmesi isteniyorsa testler sırasında çıkan tüm hataların kapatıldığından emin olunmalıdır. Bulunan hatalar, hataların kaynağı, yapılan düzenlemeler raporlanmalıdır. Bu amaç için oluşturulacak olan örnek bir hata raporu şöyledir:



2.4.1. Hata Raporlama

Hata No	Bildiren Kişi	Hata Bildirim Tarihi
Hata Adı	**Hatanın Durumu**	**Yazan/Kodlayan**
Hata Açıklaması		
Hata Tipi	**Önem Derecesi**	**Öncelik**
Atanan Kişi	**Atanma Tarihi**	**Kapanma Tarihi**
Test Edilen Öğe	**Sürüm No**	

2.4.2. Hata Önem Dereceleri

- Test sürecinde tespit edilen hatalar raporlanırken aşağıdaki hata önem dereceler kullanılabilir.

Ölümcül: Testlerin devam etmesini engelleyecek hataları belirtmek için kullanılan derecedir. Eğer bu türden bir hata bulunmuş ise testlerin devam etmesi imkansızdır.

Kritik: Testler devam edebilir ancak bu hata derecesi ile yazılım teslim edilemez.

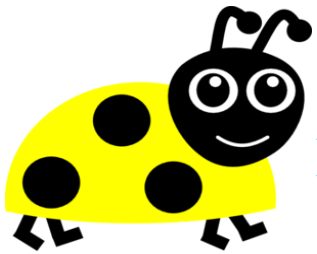
2.4.2. Hata Önem Dereceleri

Büyük: Testler devam edebilir. Ürün bu hata ile teslim edilebilir. Ancak yazılım kullanıldığında telafisi zor sonuçlar doğurabilir.

Orta: Testler devam edebilir. Ürün bu hata ile teslim edilebilir. Ancak yazılım kullanıldığında telafisi mümkün sonuçlar çıkartabilir.

Küçük: Testler devam edebilir. Ürün bu hata ile teslim de edilebilir. Yazılımın işleyişinde ortaya çıkabilecek hatalar önemli bir sonuç doğurmaz.

2.4.2. Hata Önem Dereceleri



Kozmetik: Yazılım üzerindeki renk, font, büyüklük gibi görsel hatalardır. Olması durumunda ne testi durdurur ne de ürünün teslimini engeller.

- Bu hata önem dereceleri hataların gruplandırılmasında ve testlerin başarılı olma kriterlerinde kullanılabilir.

2.5. Test Sonuç Raporlama Ve Değerlendirme

- Testler gerçekleştirildikten sonra elde edilen test verileri raporlanmalı, analiz edilmeli ve değerlendirilmelidir. Değerlendirmede test planlarında belirtilen test geçme/kalma kriterleri dikkate alınır. Test sonuçlarının raporlanması ve değerlendirilmesi için test özet belgesi hazırlanır. Test özet belgesi aşağıdaki bilgileri kapsamalıdır.

Test Özet Belgesinde Olması Gerekenler						
Modül/Birim	İhtiyaç Numarası	Test Durum Numarası	Test Sonucu Numarası	Test Sonucu	Test Tarihi	Testi Yapan
Testi İzleyen	Tekrar Sayısı	Hata Bildirimi Numarası	Sürüm No	Donanım Sürümü	Yardımcı Yazılımların Sürümü	Notlar

2.5. Test Sonuç Raporlama Ve Değerlendirme

- Test özet belgesinde test sonuçları analiz edilip değerlendirilirken kaç adet ihtiyacın, kaç adet test durumu ile kaç kere test edilerek doğrulandığı, ne tür hatalar çıktığı ve bu hataların hangi yazılım sürümünde düzeltilerek testlerden geçtiği raporlanır.

2.6. Yazılım Test Riskleri

- İyi hazırlanılmadan başlanan bir testin önce kendisi büyük bir risk oluşturur. Normal bir süreç olarak izlenen yazılımın test aşamasında da karşılaşılması olası temel riskler şunlardır:

Tümleştirmede karmaşa: Tek bir birimden oluşan yazılımın testi başka birimleri etkilemediği için karmaşıklık oluşmaz. Ancak, birden fazla yazılım biriminin tümleştirilmesi ve testi sırasında karşılıklı olumsuz etkilenmeler oluşabilir. Aynı anda test edilen birim sayısı arttıkça hataların nereden kaynaklandığını bulmak da güçleşir.

2.6. Yazılım Test Riskleri

Sıralama: Test yordamlarının belirli bir sıra takip etmeleri gerekmektedir. Uygun bir sıraya göre yapılmayan ve senaryosu iyi tanımlanmayan test durumları tekrarlara ve sonuçta da zaman kaybına yol açabilir.

Paralel test işlemleri: Birden fazla öğe aynı anda test ediliyorsa ve her bir öğe içinde birden fazla birim bulunuyorsa, yürütme sırasında birbirlerine olan etkilerinden dolayı hangi öğenin doğru, hangisinin kusurlu olduğunun bulunmasında güçlük çıkabilir.

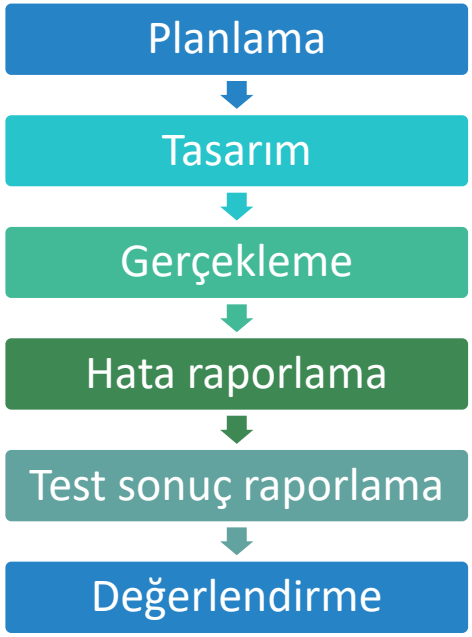
Yüksek maliyetli testler: Bazı testler çok yüksek maliyet gerektirebilir. O nedenle son derece dikkatle planlanmalı, önceden yeterli bütçe ayrılmalı, iyi değerlendirme yapılabilmesi için kayıtlara önem verilmelidir.

2.6. Yazılım Test Riskleri

Testlerin plan dışı yürütülmesi: Testlerin bir plana göre yürütülmemesi durumunda test ortamının kullanımında, testlerin uygulanmasında, hata bulmada karmaşa yaşanabilir. Plansız yapılan testlerle zaman kaybı yaşananabilir, hataların kaynaklanma nedeninin bulunması zorlaşır.

Test yordamlarının yetersizliği: Test yordamları yeterince kapsamlı olmayıp yüzeysel olarak uygulanırsa testler başarılı geçse bile atlanmış durumların gerçek kullanım sırasında sorun çıkarması beklenmelidir.

Özet



- Yazılım test süreci önceden planlanan, sonra icra edilip bildirilen ve resmi olarak belgelendirilen bir grup eylemden oluşur. Bu süreç, geliştirilen yazılımın içerisindeki olası hataların varlığına odaklanır. Bir test süreci yanda gösterilen adımlarından oluşur. Yazılım projeleri kapsamında icra edilecek olan testler projenin başında planlanmalı ve belgelenmelidir.

Çalışma Soruları

1. Yazılım yaşam döngüsü içerisinde yazılım testleri ne zaman başlar?
2. Yazılım test sürecini çizerek açıklayınız.
3. Yazılım testlerinde ne tür yardımcı araçlar kullanılır?
4. Test durumu nedir? Tanımlayınız.
5. Test durumu tanımlarken mutlaka olması gereken bilgiler hangileridir?
6. Hata yönetim yaşam döngüsünü çizerek açıklayınız.
7. Test koşturmalarında tespit edilen hatalar raporlanırken hangi hata önem dereceleri kullanılabilir? Açıklayınız.

Kaynaklar

[1] Yazılım Test Mühendisliği (1. Baskı), Rifat Çölkesen, 2010

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

Sorularınız