**Okuma Notları**

* The world needs good, focused expert testers.
* Tüm şirketlerin 3 ana bileşeni-**3P Kuralı:**

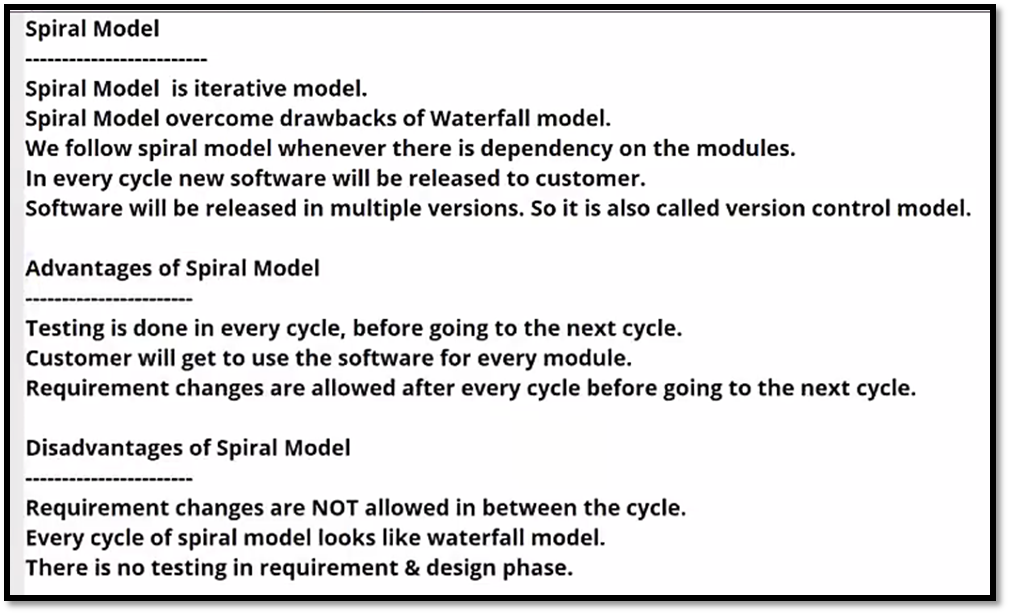
**1- People**

**2- Process**

**3- Product**

* **Project:** Müşteriye özel /sadece bir şirketin ihtiyaçları ve kullanımı için) hazırlanır. Project Based Companies denir bunlara.
* **Product:** Piyasa ihtiyacına göre genel kullanım için hazırlanır. Product Based Companies: Microsoft, Google vb. herkese açık, herkesin kullanabileceği ürünler yapıyor.
* **SDLC:** Software Development Life Cycle (Requirements, design, code-develop, test, deploy). Test aslında sonuçta değil süreç boyunca kalite artırımı için sürekli yapılır.





**V-MODEL TESTING**

**Her aşamada test yapılır.** (Waterfall da sadece belirli bir aşamada terster devreye girer. Burada her aşamada bulunur).

1. **Çok fazla döküman üretilir.**
2. **Başlangıca çok yatırım yapmak gerekir.**

* **STLC:** Software Testing Life Cycle (Finding defects-bugs, reporting defects-bugs etc.)
* **Unit testing** (kod bloklarının kendi içinde tutarlılığı) and **integration testing** (blokların bir arada uyum içinde çalışması ve desig documentlara göre testing) **bizzat developer/programmer** tarafından yapılır (logic test yaparlar). Bunlara **White Box-WB testing** denir.
* Testerların yaptığı testler: **“System testing”** (complete test). Kullanıcının requirement lerini alır ve buna uygun test planı ve **test case** (adım adım aksiyonları hazırlar) yapar.

Test case techniques test case sürecini kolaylaştırır. Örneğin, BVA (Bounder value analysis) bir username için belirlenen sınır-karakter için sınır dışına taşan bir durum var mı, istenmeyen uzunlukta ve karakterlerle username kabul ediliyor mu diye bakar. Bir kutucuğa yaş girildiğinde saçma sapan sayıların ya da sayı dışındaki şeylerin bu bölüme yazılamaması lazım. Decision (cause-effect) table tekniği sayesinde hesapta yetersiz bakiye varken ödemenin yapılıp yapılamadığını test eder. Yani if-else-whiile yapılarının düzgün çalışıp çalışmadığını kontrol etme tekniğidir. Tabii ki en iyi teknik **deneyimdir (error guessing).** Daha önceki benzer, kronik hatalar yol gösterici olur.

Ürünün kullanıcı isteklerini karşılayıp karşılamadığına bakar. Bir de **User Acceptance Testing** var. Kullanıcılar kullanırken aynı zamanda test etmiş olurlar. Bunlara **Black Box-BB testing** denir.

* Kodlama bitmeden önce yapılan testler **Verification** (Static testing-more focus on documentation) dır. .Kodlama bittikten sonra yapılan testler **Validation** (Dynamic testing-more focus on software) dur.
* **QA:** Quality assurance-süreç odaklıdır, süreçte yapılacakları define eder-belirler, her aşamada test yapılır, defect-bugs dan koruma-prevention activity amaçlıdır
* **QC:** Quality control-süreçte yapılacakları uygular, defect-bugs detection yapılır esas amaç detect activity dir, yani actual testing software dir. Kod kullanmadan manuel test yapar.

**Koddaki hata error olarak tanımlanır, tester tarafından bulunan bu error defect olarak tanımlanır, designer/developer tarafından kabul edilen bu defect'e ise bug denir.**

* **Manuel testers ne yapar?**

**1- Understand the requirements or expectations**

**2- Write test cases**

**3- Execute test cases**

**4- Find bugs-defects and report them to developer**

**5- Participate different review meetings**

Manual testingde tester ın bilgisi, yaratıcılığı, deneyimi ve analitik yetenekleri önemlidir. Sezgi ve deneyim önemli olduğundan makineler tarafından yapılamaz. İnsandan kaynaklı olarak zaman alıcıdır, hatalara daha açıktır, test edilecek hususlar sınırlıdır. Ucuzdur. Tek başına yeterli değildir. Otomasyonla birlikte daha güçlü olur.



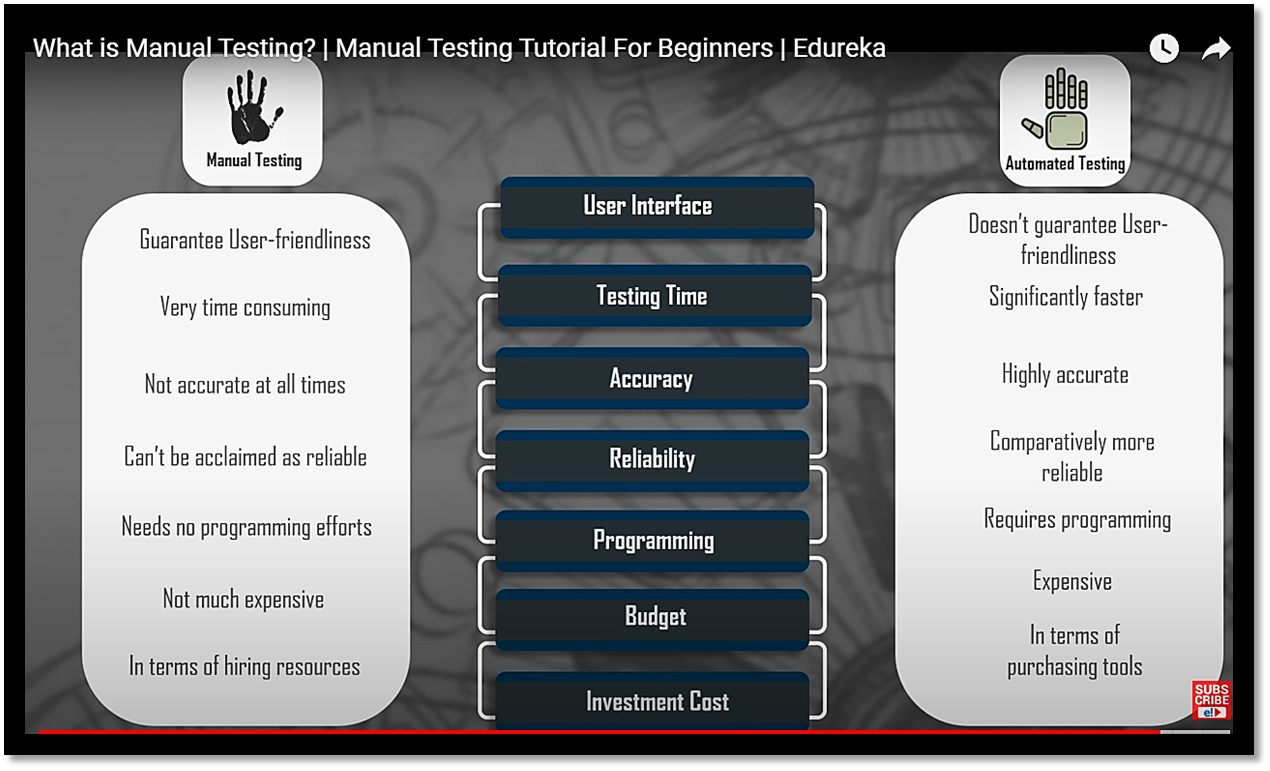
* **QE:** Quality engineering. Kodlama kullanarak otomasyon testi yaparlar.
* **Automation testers ne yapar?**

**1- Write automation scripts**

**2- Execute automation scripts**

**3- Generate reports**

**4- Status reports send to management team**



* **Software Quality Parameters:**

**1- Bug free**

**2- Delivered on time**

**3- Within Budget**

**4- Meet requirements or expectations**

**5- Maintainable (User friendly)**

* **Why the software has bugs:**

**1- Miscommunication or no communication** (Developer ve tester arasındaki iletişim sorunları, firma ihtiyaçlarına uygun hareket edilmemesi)

**2- Software complexity**

**3- Programming errors**

**4- Changing requirements** (Firma ihtiyacı yazılım aşamasında sürekli değişirse uyum sağlamak zorlaşır ve yamalı bohçaya döner, defects ihtimali artar)

**5- Lack of skilled testers** (Sürekli olumsuz düşünmeli, eksik bulmaya odaklanmalı)

**Static Testing Nedir?** (Verification aşamasında yapılır. **Verification-**Doğru ürünü mü inşa ediyoruz? Belgelere odaklanır )

Testing the project related documents is called as static testing (yazılımı değil sadece proje dökümanlarını inceliyoruz)

1. **Review** (Belge odaklıdır, belgenin doğru ve tam olup olmadığına odaklanılır. Design, code test plan ve cases review edilir. Bir kişi yapar)
2. **Walkthrough** (Informal review türüdür. Yani plansız yapılır, ihtiyaç halinde göz gezdirilir, ilgili kişilerle ayak üstü yapılır. Çok kişi yapar)
3. **Inspection** (En resmi incelemedir, 3-8 kişilik toplantılarda okuyan, yazman ve moderator bulunarak konu masaya yatırılır. Neticeler belli program dahilinde ilgili herkese duyurulur)

**Dynamic Testing Nedir?** (Validation aşamasında yapılır. **Validation**-Ürünü doğru şekilde mi inşa ediyoruz? Yazılıma odaklanır)

Testing the actual software.

1. **Integration testing ()**
2. **Unit testing ()**
3. **System Testing (**Müşterinin ihtiyaçlarının karşılanıp karşılanmadığına odaklanır**)**
4. **User Acceptance Testing ()**

**White Box Testing Techniques** (Bu teknikleri developers kullanır, modelin nasıl çalıştığını, program dilini en iyi onlar bilir, kutunun içinde ne olduğu bilindiği için zaten adı beyaz kutudur. Bunlar ilk aşama testleridir. Program dilini bilmeyen bunları yapamaz)

1. **Integration testing**
2. **Unit testing**

**Black Box Testing Techniques** (Bu teknikleri tester lar kullanır, integration testleri bitince devreye tester ekibi girer. Kutunun için de ne olduğu tam olarak bilinmediğinden tester için bu kutu siyahtır.)

1. **System Testing**
2. **User Acceptance Testing (**Tester ve user tarafından birlikte yapılır)

***QA Vs QC***

**QA:** Process(süreç) le ilgilidir. Yazılım sürecinin her aşamasıyla (SDLC-Requirement analysis, design, coding, testing, deployment, maintenance aşamalarınin tümüyle) ilgilidir. Processoriented.

**QC:** Yazılımın test edilmesidir. Yukarıdaki SDLC aşamalarının sadece test bölümüyle ilgilidir. Product oriented.

Her şirkette **3-P** vardır. Bunlar: **People, process, product**.

**Process = QA:** Kalitenin inşa edilmesine odaklanır. Defects lerden korunmak için uğraşır.

**People = QC = Testers:** Kalitenin test edilmesine odaklanır. Defects leri bug ları bulmak için uğraşır.

**Testin Aşamaları:**

**1-** **Unit testing (**Ünitelerin yani yazılımdaki küçük parçaların-single component of software- test edilmesidir. Döngü-koşul vb yapıların düzgün çalışıp çalışmadığını teker teker control eder. Developer yapar**)**

**2-** **Integration testing (**Bir araya getirilen parçaların, yani birleştirilen ünitelerin testidir. Parçaların-modüllerin birbiriyle uyumlu çalışmasıyla ilgilidir. Zincire her halka eklendikçe veri akışına bakılır ve test edilir, buna incremental integration test denir. Non-incremental integration testing yöntemi de vardır. Parçalar bir kerede eklenir. Tüm zincire tek seferde bakılır. Developer yapar**)**

**3-** **System Testing (**Uygulama fonksiyonel mi, müşteri ihtiyaçlarını karşılıyor mu? Yani bu testi yapabilmek için Customer Requirements mutlaka bilinmelidir. Terter lar tarafından yapılır.

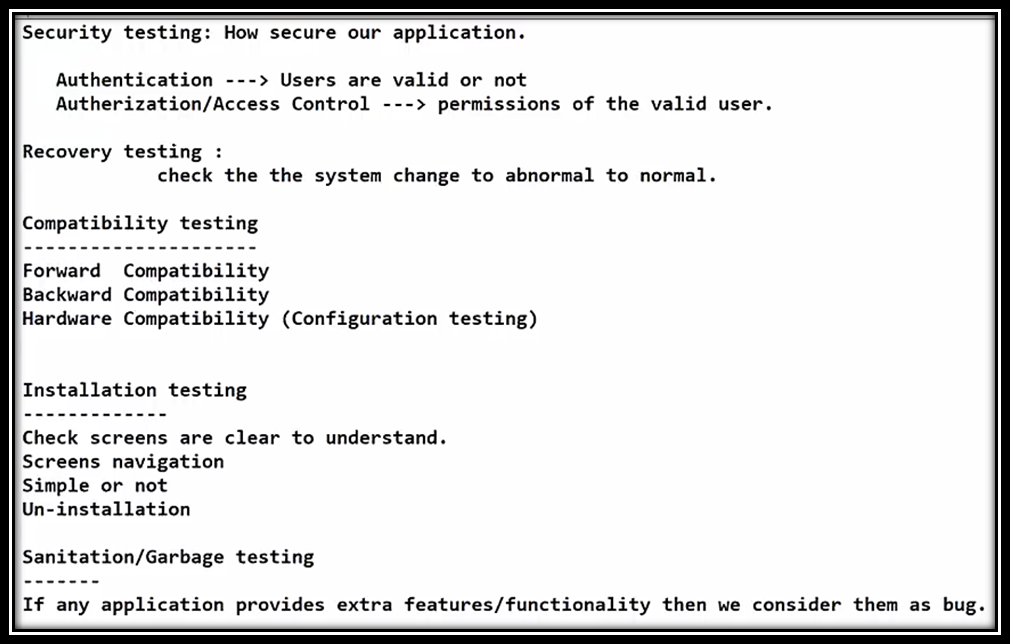
**GUI** (Graphical user interface-color, font, size, icon, images, menus gibi görünüme dair-fronten de ilişkin her şey kontrol edilir). Kısacası insert-delete-update-select operations çalışıyor mu? Bu çok detaylı bir listeye sahip. Linkedin den bulduğum listeye bak.

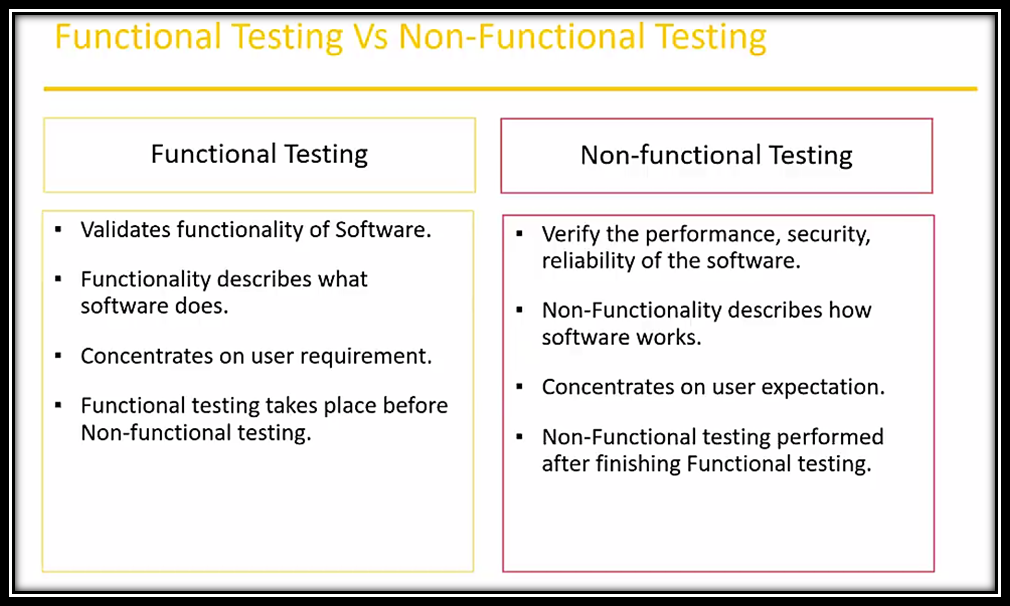
**Usability Testing**-Kullanıcı takıldığı yerlerde yönlendirmeye ya da ayrıca yardım alabileceği bir butona sahip mi, kullanıcı dostu mu, help bölümleriyle sorunların üstesinden gelebiliyor mu?

**Functional Testing**-Kullanıcının ihtiyaçlarıyla ve uygulamanın davranışlarıyla ilgilenir, özelliklerin çalışığ çalışmadığına odaklanır. Veri girilecek kutucukların düzgün çalışmasını-enable or disable durumları, seçeneklerin tıklanıp tıklanamamasını veya veri girişi yapılıp yapılmamasını test eder. Error mesajları okunabilir-kullanıcı tarafından kolay anlaşılabilir olmalıdır. Bunlar bu aşamada test edilir. Hesap yapılacak bölümler varsa işlemlerin doğru olup olmadığı kontrol edilir. **Cookies-kullanıcı tarafı&Session-server tarafı** kontrolü yapılır. Cookies browser tarafından kaydedilen geçici bilgilerdir. Daha önceki giriş ve hesap bilgileri sonrasında girişi kolaylaştırmak için kaydedilir.

**Non-Functional Testing**- Kullanıcının beklentileriyle ilgilenir. Son aşamadır. Fonksiyonel bir sorun yoksa sistemin performansının-hızının-yükleme sürelerinin, uyumunun ve güvenliğinin optimum seviyede olup olmadığına bakılır.

* **Load testte** kullanıcı sayısı kademeli artırılır ve hız gözlenir.
* **Stress testi**nden kullanıcı sayısı birden bire artırılır ve sistemde ani bir kullanıcı artışı olduğunda ne yaşandığını gözler. **)**

**4-** **User Acceptance Testing (**System testleri tamamlandıktan sonra yapılır. **Alpha**-user bile olsa şirkete gelir ve orada yapar ve **Beta Testing**-user kendi ortamında uygulamayı kurar ve test eder olmak üzere 2 çeşittir. Tester ve user lar beraber yapar**)**

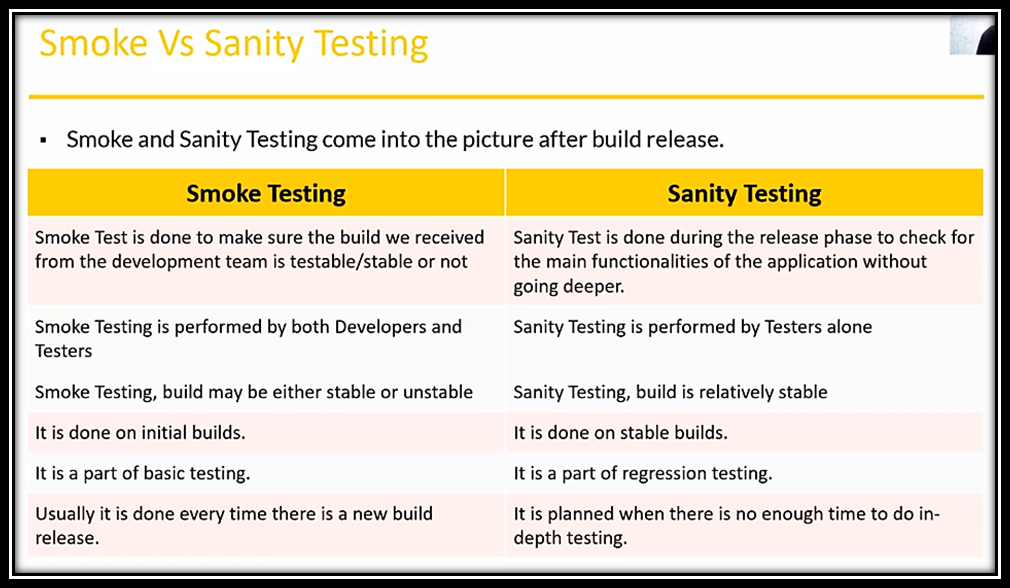


**Regression Testing**

Birçok modülden oluşan yazılımın bağlantılarının analizidir. Ekleme-çıkarma gibi değişikliklere maruz kalmış yazılım yapısının modülleri arasındaki uyumun düzgün çalışıp çalışmadığını test eder. Önceden bug sebebiyle yapılan değişikler sonrasında sorunlu parçaların ya da bütünün düzgün çalışıp çalışmadığını test eder.

**Re-Testing**

Yazılımdaki defect tespit edilir, developer bugs ları tamir eder, tester tamir edilen bugları tekrar inceler. Bu döngü sağlıklıysa süreç tamamlanır. Değilse hatalardan kurtulana kadar döngü şeklinde devam eder.

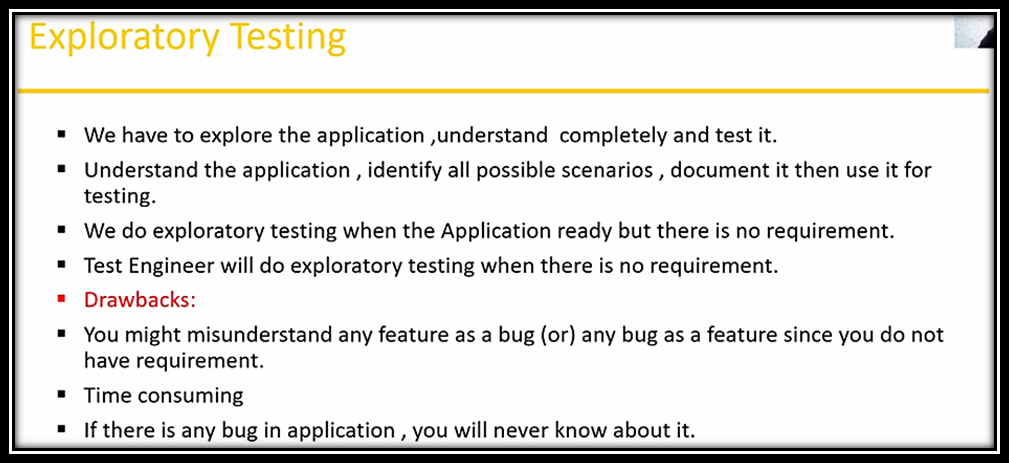


**Sanity Testing:** Yazılım düzgün şekilde açılıyor mu, temel fonksiyonlar, log-in, log-out özellikleri çalışıyor mu diye basic features ve functions a bakar. Bir yazılımda son gün yarın ve hiç testten geçmedi, test de yapmak gerekiyor. Bu durumda Sanity test yapar basic functions a bakılır. Yani en sonda, yumurta kapı saikiyle yapılır.

**Smoke Testing:** Ta en başta ortada adam akıllı yazılım yapısı yokken başlar. Her aşamada yazılımın oturmasını sağlamaya çalışır.

**Aşağıdaki 3 test birbirine çok benzer. Bunlar “exploratory, AdHoc, Monkey-Gorilla” testleridir.**

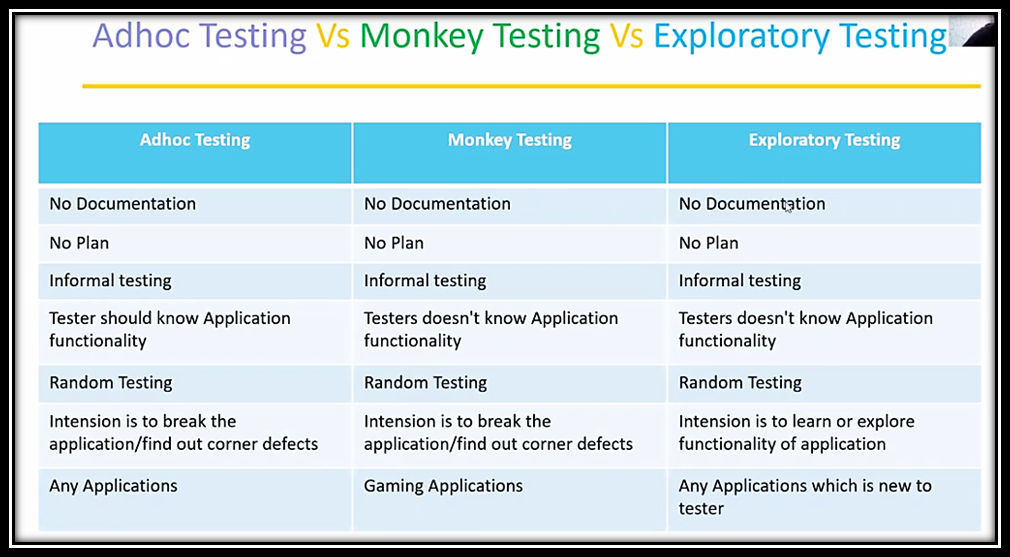
**1- Exploratory Testing-Keşif, İstikşafi Testler**: Elde yeterince requirement bilgisi yokken, test case yokken el yordamıyla yapılır. Uygulama incelenir, ortaya bir test belgesi çıkarılmaya çalışılır, ihtimalleri içeren bu belgeyle işe başlanır. Yazılıma ait ihtiyaç-gereksinim listesi eksikliği sebebiyle bazı drawbacks-eksiklik-mahsur-dezavantaj vardır. İhtiyaç-gereksinim listesi eksikliği sebebiyle özellikler bugs, buglar da özellik zannedilebilir. Zaman alıcıdır, bazen bugs hiç farkedilmeyebilir. Amaç eksik-defect bulmak değildir, keşiftir, fonksiyonları öğrenmedir.

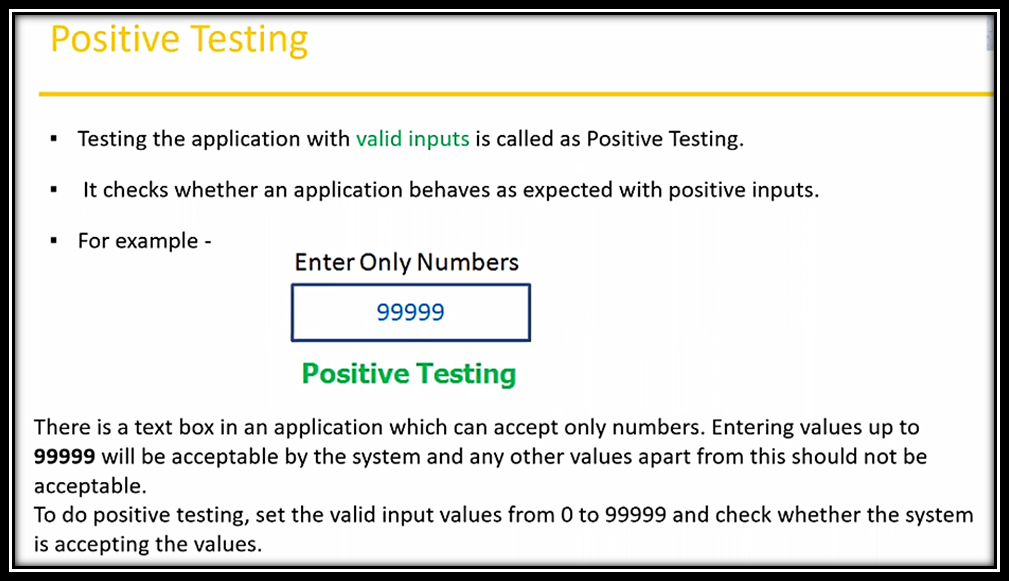


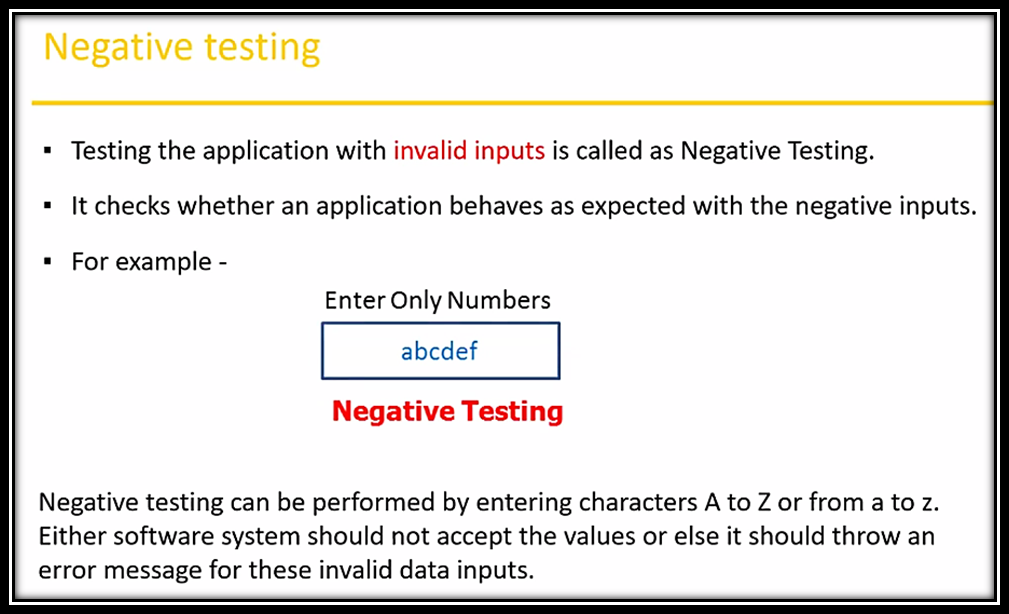
**2- AdHocTesting-Anlık, plansız testler:** İnformal dir. Teste dair başlangıç belgeleri-business test requirement-test design-test case olmadan yapılır. Randomly, plansız ve programsızdır. Tester tecrübesiyle belge olmadan ne yapması gerektiğini, uygulamanın fonksiyonelliğini bilmelidir. Amaç eksik-defect bulmaktır.

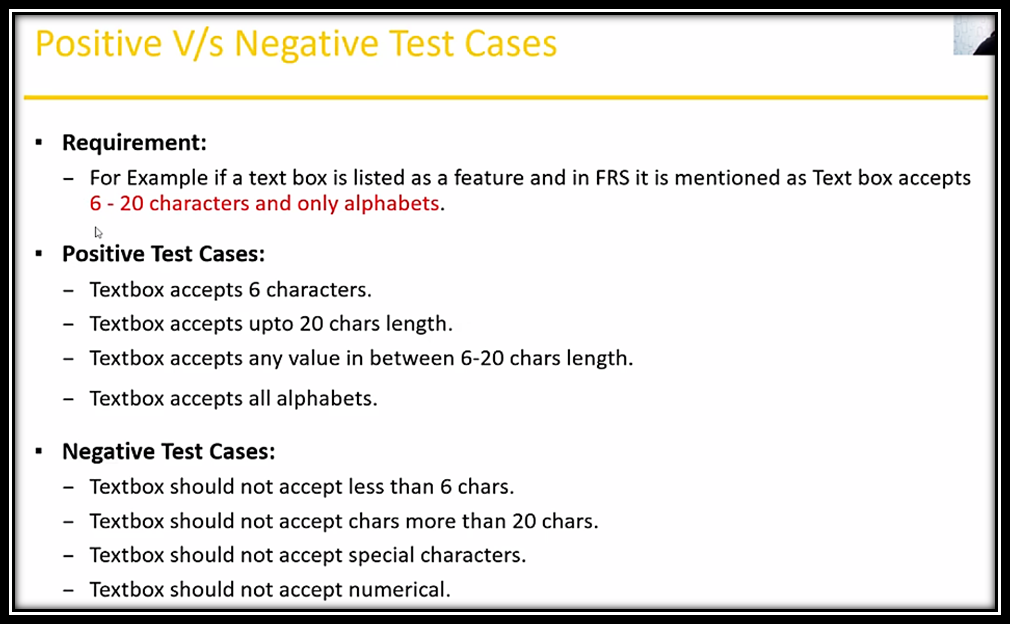
**3- Monkey-Gorilla Testing:** İnformal dir. Bu dateste dair başlangıç belgeleri-business test requirement-test design-test case olmadan yapılır. Yazılım hakkında hiç bilgi yoktur, stable değildir. Oyun yazılımlarında daha çok kullanılır. Amaç eksik-defect bulmaktır.

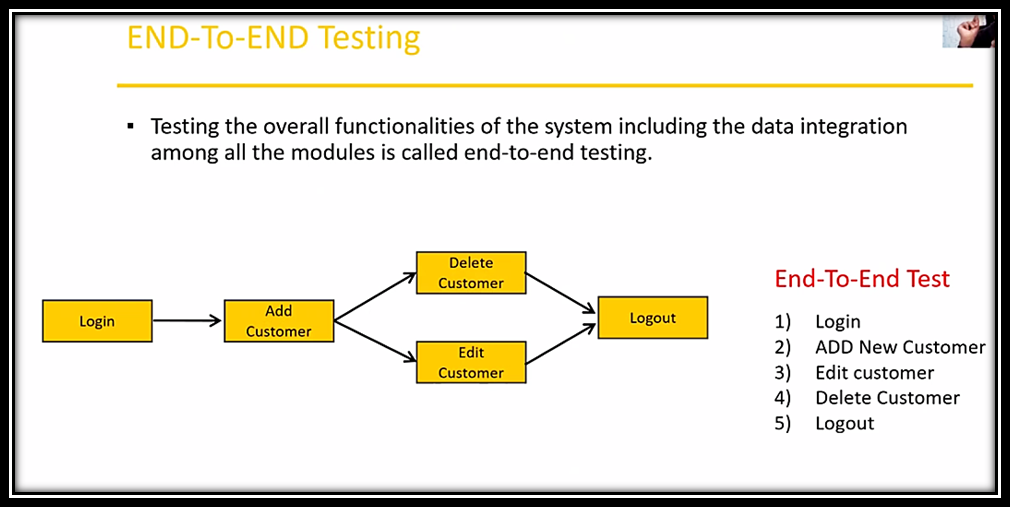
**Üç Testin de benzer tarafları:** Plansızdır. İnformal dir. Belgesiz işe başlanır. Üçü de random testidir.









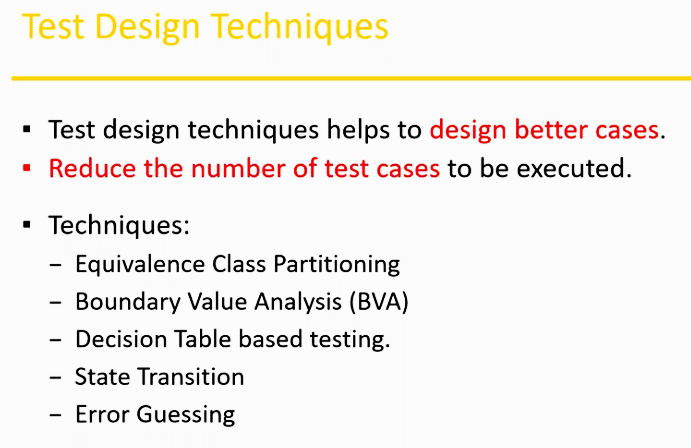


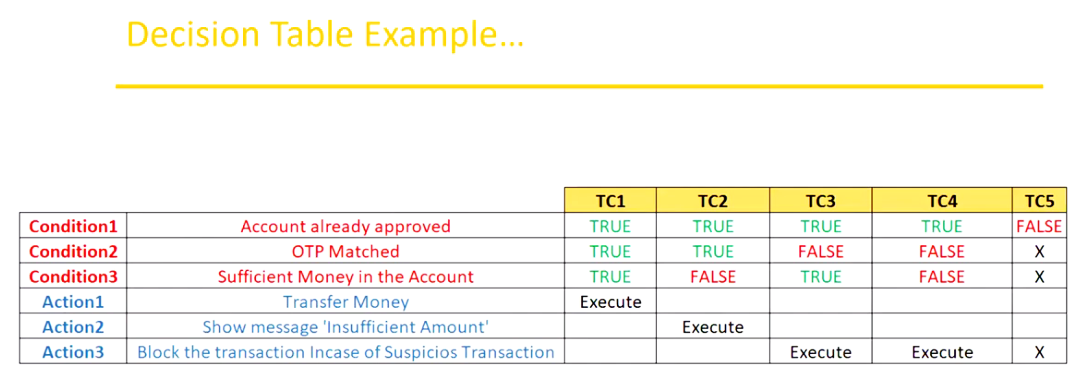
**Internationalization Testing = I 18 N Testing**-I harfinden sonra 18 karakter en sonda da n harfi olduğundan aşağıdaki testlere bu isimde verilebilir: **I18N Testing**

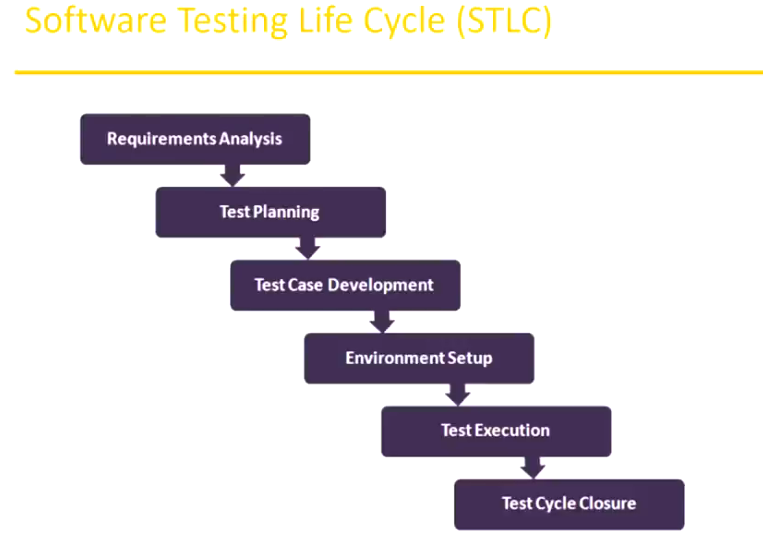


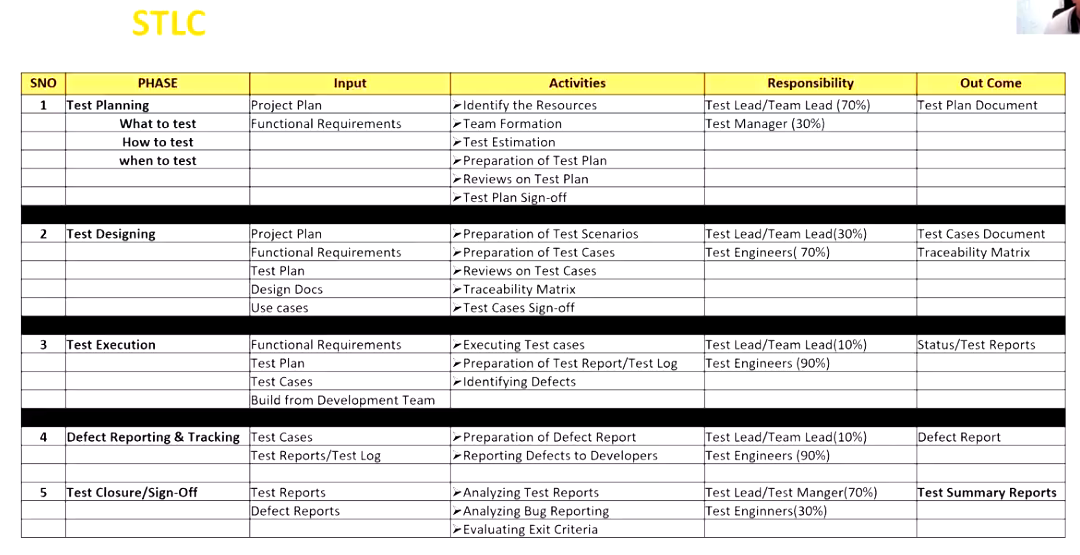
**Testing Design Techniques**

Bu teknikler veriyi azaltır, kapsamı artırır. Mesela BVA’da sadece sınırın hemen ötesindeki mak-min değerlere ve sınırlara bakılır, bunlar tutuyorsa arası zaten sağlar ve herşeye bakılmadan test edilmiş olur.





**STLC**



**Bug free yazılım yoktur. Defects 3 türlüdür:**

**P-1:** Very High Level

**P-2:** Major

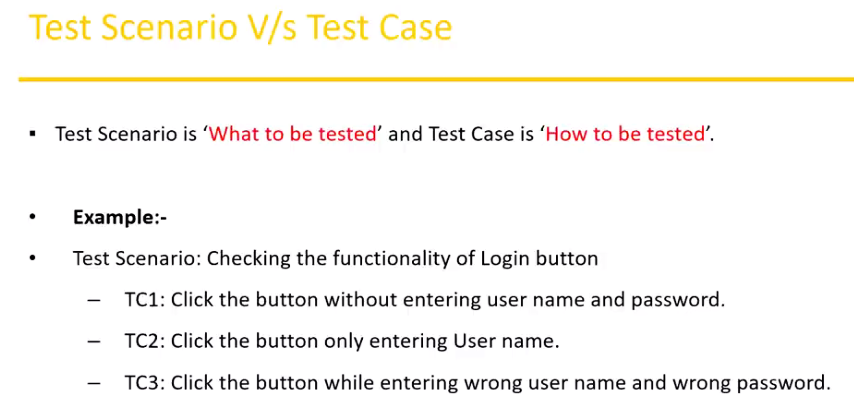
**P-3:** Minor

P1 ve P2 mutlaka düzeltilir. Yazılımlarda P3 ler bulunabilir. Bunlar daha sonra kullanıcılar tarafından tespit edilse de ana fonksiyonları etkilemediğinden başlangıçta düzeltilmeden bırakılabiliyor.

**Use Case:** Business analyst-Product manager tarafından hazırlanan, requirement belgelerinin bir parçasıdır. Olması gereken fonksiyonların listesidir.

**Test Scenario:** “What to test” sorusuna odaklanır. Neler test edilecekse listelenir.

**Test Case:** “How to test” nasıl test edilir sorusuna odaklanır. adım adım use case de belirlenen fonksiyonların nasıl test edileceği belirtilir. Tüm aşamalara, beklenen ve gerçek sonuçlara yer verilir. Beklenen sonuçlar çıkmazsa gerçek sonuçlarla-beklenenler karşılaştırılır ve uyumsuzluklar defect olarak adlandırılır.



**BUG-DEFECT-ISSUE**

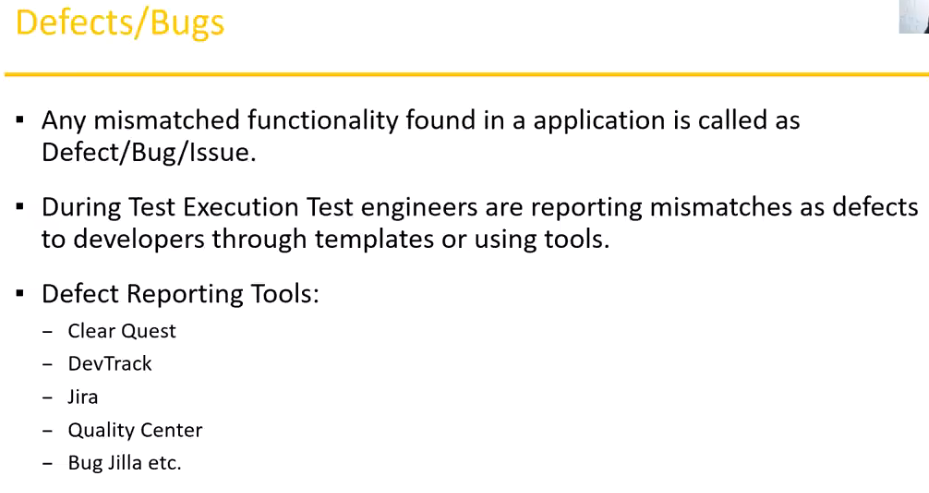
“Query findings ve issue” kavramları daha flue kavramlar. Kusur veya ayıp anlamına gelmiyor.

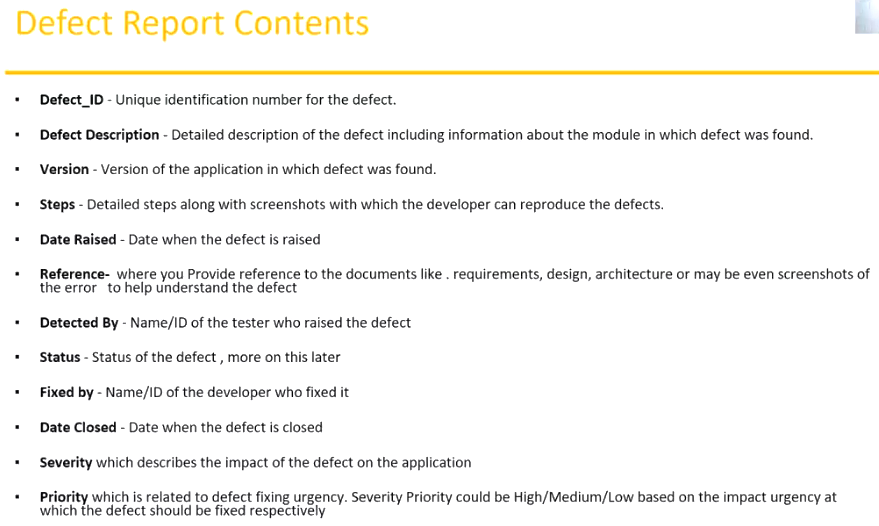
Defect-Bug ise daha iddialı (beklenen ve gerçekleşen durum arasındaki uyumsuzluklar, mismatched activities) sert kavramlar.

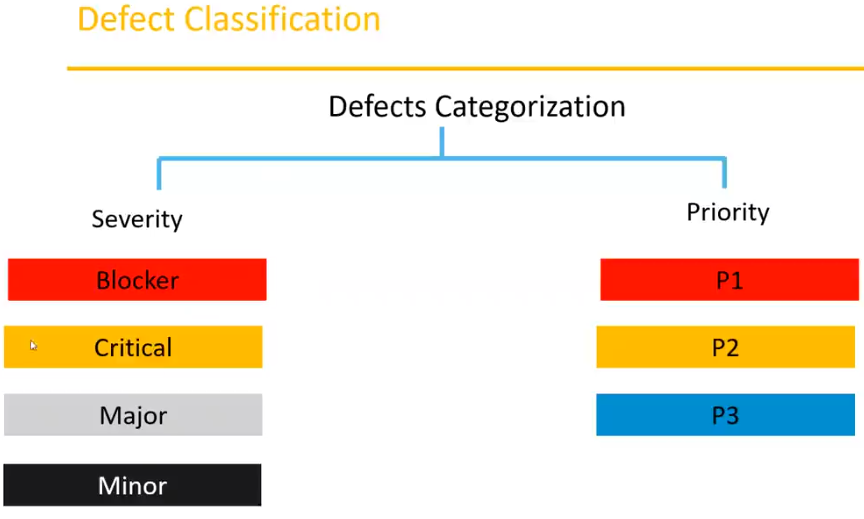
Error-mistake doğrudan insan hatalarıdır.

**Test Management Tools:** Jira, Quality Center. Bunların kapsamı geniş. Test senaryoları, cases akla gelebilecek tüm aşamalar bu tool üzerinde tanımlanır, üzerine bug reporting de yapılabilir. Jira Agile based bir tooldur. Hem developer hem tester Jira yı kullanır.

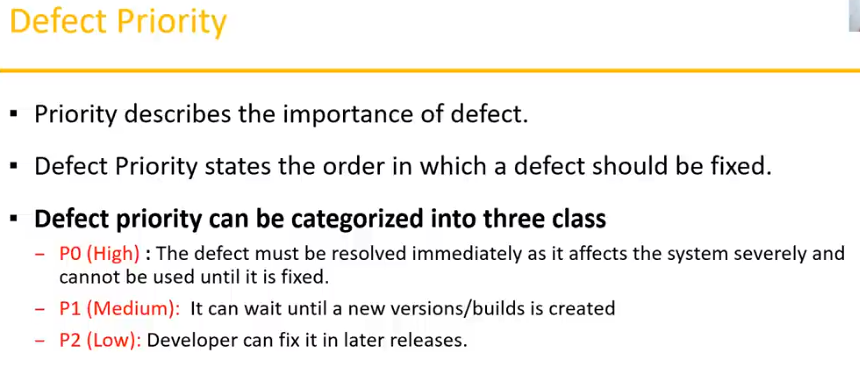
**Bug Tracking Tools:** ClearQuest, DevTrack, Bug Jilla. Bunlar sadece bug reporting amacıyla kullanılabilir.



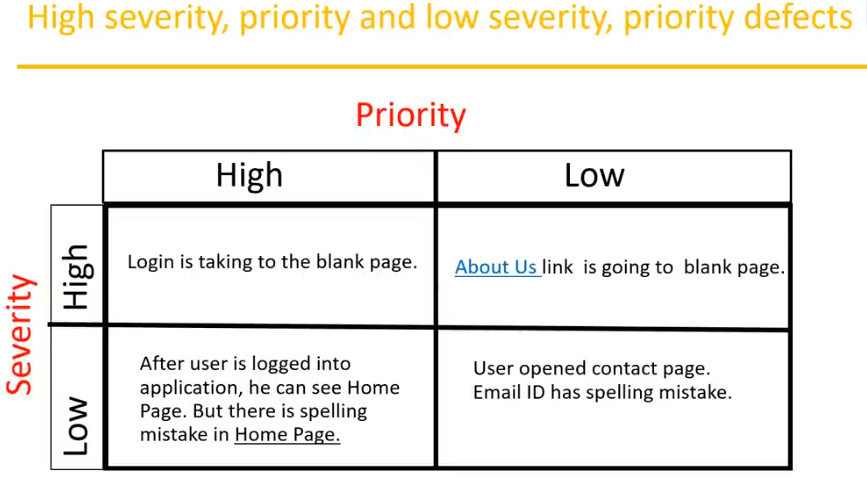


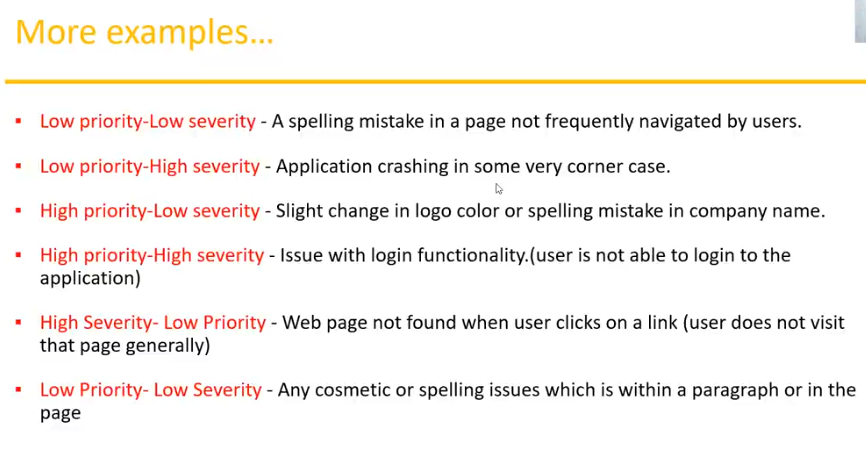




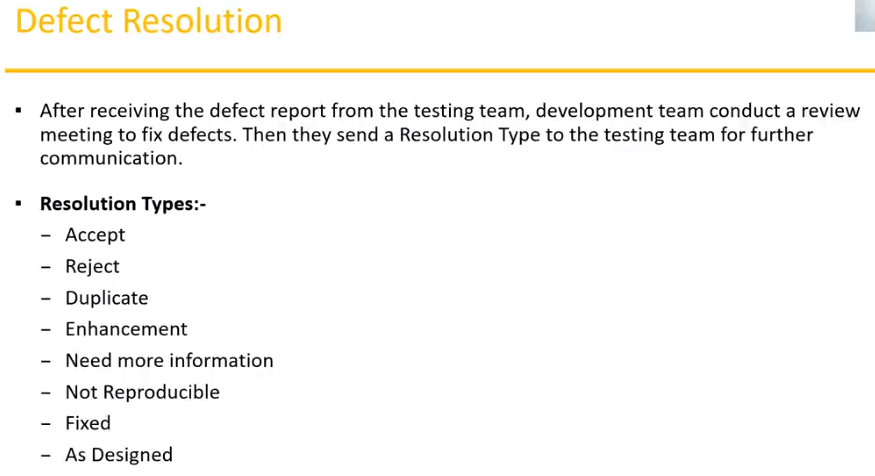


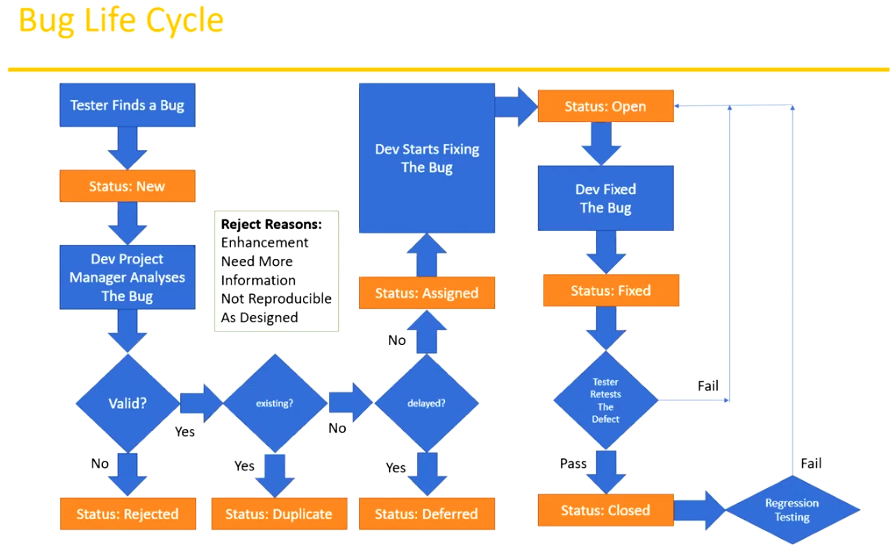
* **Severity:** Seriousness of the defect. Buna mutlaka tester karar verir. Çünkü testin tüm aşamlarına karar veren odur. Değişiklik olmaz. Product manager, developer bile buna karışmaz.
* **Priority:** How soon the defect should be fixed. Bunu da tester gruplandırır ama listeler. Değişiliğe açıktır. Developer, business analyst, product manager gibi kişiler bu önceliği değiştirebilir ya da değiştirmeyebilir. Müdahale hakları var sonuçta. Belirlenen sıraya göre bulgular düzeltilir, ele alınır.

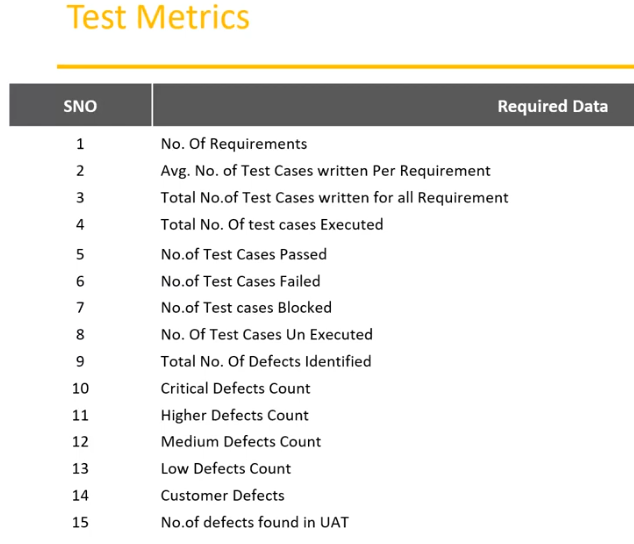




* **Defect Resolution:** Tester ın bulguları developer tarafından aşağıdaki şekilde ayrıştırılır, bulguları aşağıdaki şekilde yorumlar. Bizim rapordaki bulgulara developer aşağıdaki işlemlerden birini yapar. En alttaki “as designed” ın anlamı: tester ın defect zannettiği şey aslında yazılım dizaynının bir parçasıdır. “Duplicate”: daha önce benzer bir defect var demektir. “Enhancement”: Bazen feature defect gibi görülebilir. Developer bizim bulguya bakar, bu sonraki sürümde koyacağımız bir feature olacak, gerçek bir bug değil, güncellemelerle zaten bu özellik konulacak der.







* Testlere hazırlık yapmak uygulamaktan daha uzun sürer. Başlangıç dökümantasyonu çok önemlidir. Bu çerçeve düzgün hazırlanmazsa birçok defect gözden kaçabilir. Her case sürekli güncel ve amaca uygun olmalı.
* Uygulamaya özel test case oluşturmalıdır. Tek bir kalıp test yöntemi yoktur.

