ISTQB İÇİNDEKİLER TABLOSU

ISTQB NOTLARI (20 OCAK 2021)

K1: hatırla
K2: anla
K3: uygula

CHAPTER 1

1. Yazılım Testinin Temelleri

Yazılım Testinin Temelleri için Öğrenme Hedefleri:

kapsam, hata ayıklama, hata, insan hatası, arıza, kalite, kalite güvence, kök neden, test analizi, test esası, test senaryosu, test tamamlama, test koşulu, test kontrolü, test verileri, test tasarımı, test koşumu, test koşum çizelgesi, test uyarlama, test gözetimi, test nesnesi, test hedefi, test sonucunu bilen, test planlaması, test prosedürü, test grubu, test, test yazılımı, izlenebilirlik, sağlama, doğrulama

Keywords
coverage, debugging, defect, error, failure, quality, quality assurance, root cause, test analysis, test basis, test case, test completion, test condition, test control, test data, test design, test execution, test implementation, test monitoring, test object, test objective, test oracle, test planning, test procedure, test process, test suite, testing, testware, traceability, validation, verification

1.1. Yazılım Testi Nedir?

Yazılımlar, iş uygulamalarından tutun da (örneğin bankacılık) tüketici ürünlerine kadar (örneğin arabalar) yaşamın ayrılmaz bir parçasıdır. Çoğu insan, beklendiği gibi çalışmayan bir yazılım ile karşılaşmıştır. Düzgün çalışmayan yazılımlar; para, zaman veya ticari itibar kaybetme ve hatta yaralanma veya ölüm gibi birçok soruna yol açabilir. Yazılım testi, yazılımın kalitesini değerlendirmenin ve kullanım sırasında oluşabilecek yazılım hatası riskini azaltmanın bir yoludur.

Yazılım testi hakkındaki yaygın yanlış kanılardan biri yazılım testinin yazılımı çalıştırmaktan ve sonuçları kontrol etmekten ibaret olduğudur. Bölüm 1.4'te de açıklandığı gibi, yazılım testi birçok farklı faaliyeti içeren bir süreçtir; test koşumu aktivitesi (sonuçların kontrolü de dâhil olmak üzere) bu faaliyetlerden sadece bir tanesidir. Test süreci aynı zamanda test planlama, test analizi, test tasarımı, testin uyarlanması, test ilerlemesini ve sonuçlarını raporlama ve bir test nesnesinin kalitesini değerlendirme gibi faaliyetleri içerir.
Bazı testler, test edilen birimin veya sistemin çalıştırılmasını içerir; bu testlere dinamik testler (dynamic testing, dynamische test) denir. Diğer testler, test edilen birimin veya sistemin çalıştırılmasını gerektirmez; bu testlere ise statik testler (static testing, statische Tests) denir. Dolayısıyla testler; gereksinimler (requirement, Anforderungen),

kullanıcı hikâyeleri kullanıcı hikâyeleri(user stories, User-Stories) ve kaynak kod(source code, Quellcode) gibi çalışma ürünlerinin gözden geçirilmesini de içerir.
Testler hakkındaki bir başka yanlış kanı ise testlerin tamamen gereksinimlerin doğrulanmasına, kullanıcı hikâyelerine veya diğer spesifikasyonlara odaklandıklarıdır. Her ne kadar yazılım testleri sistemin belirli gereksinimleri karşılayıp karşılamadığını kontrol etmeyi içerse de, aynı zamanda sistemin operasyonel ortamında/ortamlarında kullanıcı ve diğer paydaş ihtiyaçlarını karşılayıp karşılamadığını kontrol etmek anlamına gelen sağlama yapmayı da içerir.
Test faaliyetleri farklı yaşam döngülerinde farklı şekilde düzenlenir ve yürütülür (bkz. Bölüm 2.1).

(Düzgün çalışmayan yazılımlar; para, zaman veya ticari itibar kaybetme ve hatta yaralanma veya ölüm gibi birçok soruna yol açabilir.
Yazılım testi, yazılımın kalitesini değerlendirmenin ve kullanım sırasında oluşabilecek yazılım hatası riskini azaltmanın bir yoludur.
Test süreci(Test process,) aynı zamanda test planlama(test planning), test analizi(test analyszing), test tasarımı(test design), testin uyarlanması(implementing test), test koşumu(test executing), test ilerlemesini ve sonuçlarını raporlama(test reporting), sonuçların kontrolü(checking results) ve bir test nesnesinin kalitesini değerlendirme(evaluating the quality) gibi faaliyetleri içerir. (8 adet)
Bazı testler, test edilen birimin veya sistemin çalıştırılmasını içerir; bu testlere dinamik testler(dynamic testing, dynamische test) denir. Diğer testler, test edilen birimin veya sistemin çalıştırılmasını gerektirmez; bu testlere ise statik testler(static testing, statische Tests) denir. Dolayısıyla testler; gereksinimler(requirement, Anforderungen), kullanıcı hikâyeleri(user stories, User-Stories) ve kaynak kod(source code, Quellcode) gibi çalışma ürünlerinin gözden geçirilmesini de içerir.

Verification : Döküman üzerinde yapılan testte (Static Testte), hazırlıklar düzgün yapılmış mı yapılmamış mı, bunun doğrulanması işlevine denir.

Validation : kodu çalıştırdıktan sonra (Dinamik Testte) ortaya çıkan sonuçta beklenen netice alınıyor mu, alınmıyor mu, bunun doğrulanması işlevine denir)

1.1.1 Yazılım Testinin Genel Hedefleri (*ve Amaçları*)

Gereksinimler(requirement), kullanıcı hikâyeleri(user story), tasarım(design) ve kod(code) gibi çalışma ürünlerini değerlendirmek. (bu aşamalarda oluşabilecek hataları engellemek)
Belirtilen tüm gereksinimlerin yerine getirilip getirilmediğini doğrulamak.
Test nesnesinin eksiksiz olup olmadığının ve kullanıcıların ve diğer paydaşların beklediği şekilde çalıştığının sağlamasını yapmak.
Test nesnesinin kalite seviyesi hakkında güven oluşturmak.
Arızaları(Failures) ve hataları(defects) tespit etmek.
Özellikle test nesnesinin kalite seviyesiyle ilgili olarak sağlıklı kararlar almalarını sağlamak için paydaşlara(stakeholder) yeterli bilgiyi sunmak.
Yazılımın kalitesiz olma riskini düşürmek.
Sözleşmeden kaynaklanan, yasal veya düzenleyici gereksinimlere veya standartlara uyup uymadığını kontrol etmek.

Testin hedefleri, proje bağlamına(çeşidine), test seviyesine(test level) ve yazılım geliştirme yaşamdöngüsü modeline bağlı olarak değişebilir.
Birim testi(Component Testing) sırasındaki hedeflerden biri, olabildiğince çok sayıda arıza tespit etmek, böylece bunlara neden olan hataların erkenden tespitini ve düzeltilmesini sağlamak, birim testinin kod kapsamını(test edilen kod miktarı) artırmak.

testi(Acceptence Testing) sırasındaki hedeflerden biri sistemin beklendiği gibi çalıştığını ve gereksinimleri karşıladığını doğrulamak, belirli bir zaman aralığında sistemin piyasaya sürülmesindeki riskler hakkında paydaşlara(stakeholder) bilgi vermek)

1.1.2 Yazılım Testi ve Hata Ayıklama(Testing and Debugging, Testen und Debugging)

Test etme(Testing, Testen) ve hata ayıklama(Debugging) farklı aktivitelerdir.

Testing : Testlerin yürütülmesi(executing test, Durchfuhrung von Tests), yazılımdaki hataların neden olduğu arızaları(failur) gösterebilir. (Tester'a ait.)

Debugging : Hata ayıklama ise arızaların arkasında yatan hataları bulan, analiz eden ve düzelten(hatayı fix etme) yazılım geliştirme faaliyetidir. (Developer'a ait.)

Confirmation : Daha sonra gerçekleştirilen onaylama testleri(Confirmation Test) düzeltmelerin hataları giderip gidermediğini kontrol eder. (Tester'a ait)

İnitial Test : Bazı durumlarda, test uzmanları başlangıçtaki testlerden(Initial Test) ve son onaylama testinden(Final confirmation test) sorumluyken,

Component and Component integration test : (Birim test ve birim integrasyon testi, yazılımcılar tarafından yapılır.) yazılımcılar hata ayıklama(debugging) ve ilgili birim testlerini(component test ve component integration test) gerçekleştirir. Bununla birlikte, çevik yazılım geliştirmede ve diğer bazı yaşam döngülerinde test uzmanları hata ayıklama(debugging) ve birim testlerine(component integration test) dâhil olabilirler.

(Test etme(Testing, Testen) ve hata ayıklama(Debugging) farklı aktivitelerdir. Testing testerlar tarafından yapılır, hata ayıklama, failures'ların arkasında yatan hataları bulma, analiz etme, fix etme düzeltme faaliyeti diyeceğimiz Debugging ise developerlar tarafından yapılır.

Tester bug buldugunda bir bug ticket acıyor ve bunu bir develeper a assign ediyor. Developer bu bugi bug olarak görmeyip deny edebiliyor. Bu bir bug değil diyebiliyor. Ya da bu bug daha onceden bulunan baska bir bugin aynısı oldugunu uzerinde calıstıgını soyleyip Dublicated bug diyebilir. Defer edebiliyor: uzerinde calıstıgı buglar oldugu ıcın kabul etmeyebiliyor)

1.2.1 Yazılım Testinin Başarıya Katkısı

Uygun test tekniklerinin kullanılması, bu tekniklerin uygun test seviyelerinde ve yazılım geliştirme yaşam döngüsünün uygun noktalarında uygulanması sorunlu ürün teslimlerinin sıklığını azaltabilmektedir. Bazı örnekleri şunlardır:

1. Gereksinimlerin gözden geçirilmesi veya kullanıcı hikâyesinin iyileştirilmesi süreçlerine test uzmanlarının katılması, bu çalışma ürünlerindeki hataların tespit edilmesini sağlayabilir. Gereksinimlerdeki hataların belirlenmesi ve giderilmesi, yanlış veya test edilemeyen fonksiyonalitenin geliştirilme riskini azaltır.

2. Sistem tasarlanırken test uzmanlarının sistem tasarımcılarıyla(system designers) birlikte çalışması, tarafların tasarım ve tasarımın nasıl test edileceği konusundaki anlayışını artırabilir. Bu anlayış artışı, tasarım hatalarını azaltabilir ve yapılacak testlerin erken bir aşamada belirlenmesini sağlayabilir.

3. Kod geliştirme aşamasında test uzmanlarının yazılımcılarla(developers) birlikte çalışması, tarafların kod ve kodların nasıl test edileceği konusundaki anlayışını artırabilir. Bu anlayış artışı, kod ve testlerde hata çıkma riskini azaltabilir.

4. Canlıya(release, Freigabe : ürünün piyasaya sürülmesi) alınmadan önce test uzmanlarının yazılımı doğrulaması ve sağlamasını yapması, kaçırılması mümkün arızaları(failures) tespit edebilir ve arızalara neden olan hataları giderme sürecini destekleyebilir (hata ayıklama, debugging). Bu, yazılımın paydaş(stakeholder , müşteri) ihtiyaçlarını ve gereksinimleri karşılama olasılığını artırır.

(Uygun test tekniklerinin kullanılması, bu tekniklerin uygun test seviyelerinde ve yazılım geliştirme yaşam döngüsünün uygun noktalarında uygulanması sorunlu ürün teslimlerinin sıklığını azaltabilmektedir)

1.2.2 Kalite Güvence ve Yazılım Testi(Quality Assurance and Testing, Qualitätssicherung und Testen)

Quality Assurance and Testing : İnsanlar testlerden bahsederken sıklıkla "kalite güvence" (Quality Assurance, Qualitätssicherung) (veya sadece KG)(QA) tabirini kullansa da, kalite güvence ve test aynı şey değildir, ancak birbiriyle ilişkilidir.

Quality management : Daha büyük bir kavram olan kalite yönetimi bu ikisini birbirine bağlar. Kalite yönetimi, bir kurumu kalite açısından yönlendiren ve kontrol eden tüm faaliyetleri içerir.
Kalite yönetimi(quality control) diğer faaliyetlerle birlikte kalite güvenceyi ve kalite kontrolünü de içerir.

Kalite güvence genellikle, uygun kalite seviyelerinin elde edileceğine dair güven sağlamak için doğru süreçlere bağlı kalmaya odaklanır. Doğru süreçler doğru şekilde gerçekleştirildiğinde, bu süreçlerde oluşturulan çalışma ürünleri genellikle yüksek kalitededir ve bu da hataların önlenmesine katkıda bulunur.

Retrospective meeting(Bewertungssitzungen) : Ek olarak, süreçleri iyileştirmek için geriye dönük değerlendirme (GDD) toplantılarının bulgularının doğru bir şekilde uygulanması ile birlikte, hataların nedenlerini belirlemek ve gidermek için kök neden analizinin(root cause analyses, Grundursachenanalyse) kullanılması, etkin kalite güvence için önemlidir.
Kalite kontrol, uygun kalite seviyelerinin elde edilmesini destekleyen test faaliyetleri de dâhil olmak üzere çeşitli faaliyetleri içerir. Test faaliyetleri, genel yazılım geliştirme(software development) veya bakım sürecinin (maintenance process) bir parçasıdır. Kalite güvence, tüm sürecin doğru şekilde yürütülmesini hedeflediğinden testlerin doğru şekilde yapılmasını desteklemektedir.

(kalite güvence ve test aynı şey değildir, ancak birbiriyle ilişkilidir. Test ve kalite güvence kalite yönetimindeki faaliyetlerden bir tanesidir.
Doğru süreçler doğru şekilde gerçekleştirildiğinde, bu süreçlerde oluşturulan çalışma ürünleri genellikle yüksek kalitededir ve bu da hataların önlenmesine katkıda bulunur.
Süreçleri iyileştirmek için geriye dönük değerlendirme (GDD, Retrospective meeting(Bewertungssitzungen)) toplantılarının bulgularının doğru bir şekilde uygulanması ile birlikte, hataların nedenlerini belirlemek ve gidermek için kök neden analizinin(root cause analyses, Grundursachenanalyse) kullanılması, etkin kalite güvence için önemlidir.
Test faaliyetleri, software development veya maintenance process bir parçasıdır)

1.2.3 İnsan Hataları, Hatalar ve Arızalar(Errors, Defects, and Failures (Fehlhandlungen, Fehlerzustände und Fehlerwirkungen)

(Error(Fehlhandlungen) : (insan hatası) insanların yapmış olduğu hata, mistake, developerun kod üstünde yapmış olduğu hata. developerın gözden kaçırdığı, istemeyerek yaptığı bozukluğa insan hatası **(error)** denir.

Defects(Fehlerzustände) : (hata) developerın yaptığı insan hatasından dolayı, yazılımın bozulmasına kusur(fault), yazdıkları programlar içerisinde gözden kaçırdığı her bozukluk **defect**, ya da **bug** olarak adlandırılır.

Failures(Fehlerwirkungen) : (arıza) hem developer, hem testerın gözünden kaçan hata. Kusurlu yazılımın son kullanıcıya ulaştığı anda açığa çıkan bozukluğa arıza(**failure**) adı verilir.)

(+ yerine - yazılması bir error dur, çıkan sonuc defect tir, yapılan bu hata, üzerinde çalışılan üründe bir hataya neden olabilir. veyahutta yapılan hata çalışılan ürünün bağlantılı başka bir ürün üzerinde hataya sebebiyet verebilir, eğer defect çalıştırılırsa ve farkedilmezse arzu edilmeyen bir sonuç ortaya çıkar, buna da failur denir.

İnsanların yaptığı hataların(Error) birçok nedeni olabilir, örneğin:

1. ***Zaman baskısı***
2. ***İnsani yanılma payı***
3. ***Deneyimsizlik veya yetersiz yetkinlik***

4. *Gereksinimler ve tasarım ile ilgili yanlış iletişim de dâhil olmak üzere proje ekip üyeleri arasındaki yanlış iletişim*
5. *Kodun, tasarımın, mimarinin, çözülecek temel problemin ve/veya kullanılan teknolojilerin karmaşıklığı*
6. *Özellikle sistem içi ve sistemler arası etkileşimlerin sayısının fazla olduğu durumlarda, sistem içi ve sistemler arası arayüzler hakkındaki yanlış anlaşılmalar*
7. *Yeni, henüz tecrübe edilmemiş teknolojiler*

Arıza (Failur) sadece koddaki hatalardan kaynaklanmaz. çevresel koşullar. Örneğin, radyasyon, elektromanyetik alanlar ve kirlilik. Donanım koşullarının değişmesi yazılımın çalışmasını etkileyebilir. Beklenmedik test sonuçlarının tamamı arıza değildir.

False negatives : (Yanlış negatif) bulunması gereken hataları bulamayan testlerin sonuçlarına denir.
False positives : (Yanlış pozitif) hata olarak rapor edilmesine rağmen, gerçekte hata olmayan sonuclara denir.

Yanlış negatiflerin ve Yanlış pozitiflerin ortaya çıkma nedenleri:
1. *Testlerin uygulanma şeklindeki hatalardan.*
2. *Test verilerindeki, test ortamındaki veya diğer test yazılımındaki hatalardan.*
3. *Diğer nedenlerden dolayı ortaya çıkabilir.*
4. *İnsan hatalarından)*

1.2.4 Hatalar, Kök Nedenleri ve Etkileri(Defects, Root Causes and Effects Fehlerzustände, Grundursachen und Wirkungen)

Root causes : Hataların(defects) kök nedenleri(root causes, Grundursache), hataların oluşmasına sebep olan en baştaki eylemler veya koşullardır.

Root causes analysis : Hatalar, kök nedenlerini belirlemek için analiz edilebilir, böylece gelecekte benzer hataların ortaya çıkma olasılığının azaltılması hedeflenir. En belirgin kök nedenlere odaklanan kök neden analizi(root cause analyses, Grundursachenanalyse) gelecekte önemli sayıda hatanın ortaya çıkmasını önleyen süreç iyileştirmelerini(process improvement, Prozessverbesserung) sağlayabilir.

(Root causes : Hataların(defects) kök nedenleri(root causes, Grundursache), hataların oluşmasına sebep olan en baştaki eylemler veya koşullardır
gelecekte benzer hataların ortaya çıkma olasılığının azaltılması adına kök nedenlerini belirlemek için kök neden analizi(root cause analyses, Grundursachenanalyse) yapılır. Bu analiz gelecekte önemli sayıda hatanın ortaya çıkmasını önleyen süreç iyileştirmelerini(process improvement, Prozessverbesserung) sağlayabilir)

1.3 Yedi Test Prensibi

1. **Testin amacı, yazılımda hataların olduğunu göstermektir; yazılımda hata kalmadığını ispatlamak değildir. (Testing shows the presence of defects, not their absence)**

Testler, yazılımda hataların mevcut olduğunu gösterebilir, ancak hiç hata kalmadığını ispatlayamaz. Testler, yazılımda keşfedilmemiş hataların kalma olasılığını azaltır. Yazılımda yeni hatalar bulunamasa bile bu durum yazılımın, kullanıcıların ihtiyaçlarını karşılayacağı anlamına gelmez.
(yapmış olduğumuz testin amacı hatayı bulmak.
Test hataların var olduğunu gösterir, ama uygulama içinde hata olmadığını göstermez, keşfedilmemiş hataların ihtimalini azaltır. hiç bir hata bulunamasa bile uygulama hatasızdır deme şansımız yok.
var isbat edilebilir ama yokun isbatı olmaz.)

2. **Yazılımı %100 test etmek imkansızdır.(Exhaustive testing is impossible)**

Yazılımı tüm girdi, ön koşul ve çıktıların kombinasyonları ile test etmek çok basit yazılımlar dışında imkansızdır. Geniş kapsamlı (her durumu kapsayan) testler yapmaya çalışmak yerine, testten daha çok geri dönüş almak için risk analizi, test teknikleri ve öncelikler kullanılmalıdır.
(her türlü hatayı bulmak mümkün değildir.
her şeyiyle tastamam bir test mümkün değil.
pozitif testte sınırlandırma olabilir ama negatif testte bir sınır yok, sonsuz. bu nedenle Exhaustive testing impossible-imkansız.
ihtiyacımız kadar test yapıyoruz, zaman, bütçe vs. meseleleride söz konusu)

3. **Erken test, zaman ve para tasarrufu sağlar.(Early testing saves time and money)**

Hataları erken bulmak için, yazılım geliştirme yaşam döngüsünde hem statik hem de dinamik test faaliyetleri mümkün olduğunca erken başlatılmalıdır.
Erken teste bazen "sola kaydırma" (shift left) da denir.

4. **Hatalar yazılımın belli alanlarında yoğunlaşır(Defects cluster together)**

(Bir yerde hata bulunduğu zaman, orada hata kümlerinin oluşmuş olma riski yüksek olduğundan testi buralarda yoğunlaştırmak tavsiye ediliyor. Bu risk analizleri sırasında ortaya çıkan genel bir sonuçtur. Bu developer'ın psikolojik durumu, bilgi yetersizliği vs. gibi nedenlerden olabiliyor.)

5. **Antibiyotik direnci(Beware of the pesticide paradox)**

Aynı testler sürekli tekrar edilirse, en sonunda bu testler artık yeni hatalar bulamamaya başlar. Yeni hataları bulmak için mevcut testler ve test verilerinin değiştirilmesi ve yeni testlerin yazılması gerekebilir. Yalnız bazı durumlarda örneğin otomatikleştirilmiş regresyon testlerinde antibiyotik direnci prensibi daha önce çalışan yerlerin doğru bir şekilde çalışmaya devam ettiğini göstermek amacıyla faydalı olabilir.

6. **Yazılım testi, projenin bağlamına(çeşidi, türüne), koşullarına göre değişiklik gösterir(Testing is context dependent)**

Test, farklı proje bağlamlarında farklı şekillerde gerçekleştirilir. Örneğin, **güvenlik açısından kritik bir endüstriyel kontrol yazılımı**, bir **e-ticaret mobil uygulamasından** farklı bir şekilde test edilir (*Çeşit için örnek*). Buna başka bir örnek olarak, çevik projelerde testler, sıralı yaşam döngüsü projelerindeki testlerden farklı şekilde yapılır (*Koşul için örnek*).

7. **Yeni hata bulamıyoruz başarılı bir yazılım elde ettik yanılgısı(Absence-of-errors is a fallacy)**

Çok sayıda hatayı bulup çözmenin bir yazılımın başarısını sağlayacağını düşünmek bir yanılgıdır. Örneğin, belirlenen tüm gereksinimleri titizlikle test etmek ve bulunan tüm hataları çözmek, yine de **kullanımı zor**, **kullanıcıların ihtiyaçlarını ve beklentilerini karşılamayan** veya diğer rakip yazılımlara kıyasla **daha zayıf bir yazılımın** üretilmesini engelleyemeyebilir.

1.4 Test Süreci(Test process, Testprocess)

(Her şirketin kendine özel bir test stratejisi vardır. Test stratejisi test plana göre daha geniştir, daha üst bir kavramdır.)

1.4.1 Proje Bağlamında Test Süreci (Test sürecini etkileyen faktörler)

Bir kurum için test sürecini etkileyen bağlamsal faktörler aşağıdakileri içerir, ancak bunlarla sınırlı değildir:

1. Kullanılan yazılım geliştirme yaşam döngüsü modeli ve proje metodolojileri(*Agile, Waterfal*).
2. Ele alınan test seviyeleri(high, low) ve test çeşitleri(UI,Data base, API).
3. **Ürün** ve **proje riskleri (**üretilen ürünün risk durumu, ne kadar yatırım yapılmış, hatalı üretildiğinde ne kadar bir zarar ortaya çıkabilir. can veya mal kaybına sebebiyet verebilir mi.**).**
4. Kurumun **faaliyet alanı(**ürün hangi alanda kullanılacak**)**.
5. Aşağıdakileri içeren ancak bunlarla sınırlı olmayan **operasyonel kısıtlamalar:** *(aşağıdakilerden hangisi operasyoneldir diye sorulabilir.*)

a. **Bütçeler ve kaynaklar**

b. **Süreler**

c. **Karmaşıklık**

d. **Sözleşmeden kaynaklanan ve yasal(*prosudürden kaynaklanan*) gereksinimler**

6. Kurumsal politikalar ve uygulamalar(şirketin genel yaklaşımları, kuralları, prensipleri).
7. Sağlanması gereken iç ve dış standartlar(bunlar kanuni standartlar değil, görme engelliler için bir aplikasyon üretiliyorsa bunun bir standardı vardır, bu standartlara uygun olması gerekiyor.).

(Bir proje için uygulanacak test süreci; SDLC modeline, test seviyesine, çeşidine, Ürün ve proje risklerine, Kurumun faaliyet alanına, operasyonel kısıtlamalar, Bütçe ve kaynaklar, Süre, Karmaşıklık, Sözleşmeden kaynaklanan ve yasal gereksinimler, Kurumsal politikalar ve Sağlanması gereken iç ve dış standartlara göre şekillenir.)

Aşağıdaki bölümlerde organizasyonel(*kurumsal*) test süreçlerinin genel özellikleri açıklanmaktadır:

1. Test aktiviteleri ve yapılacak işler(tasks, Aufgaben)
2. Test çalışma ürünleri (Test work products, Testarbeitsergebnissen)
3. Test esası(test basis : üzerinde çalıştığımız test ettiğimiz, doküman ve kodlar) ve test çalışma ürünleri arasındaki izlenebilirlik(Traceability, Verfolgbarkeit)

(organizasyonel(kurumsal) test süreçlerinin genel özellikleri; Test activites, tasks(yapılacak işler), work product, Test base(üzerinde çalıştığımız test ettiğimiz, doküman ve kodlar) ve Traceability(izlenebilirlik))

Test esasının(Test bases) (ele alınan her test seviyesi veya çeşidi için) ölçülebilir kapsama kriterlerinin(measurable coverage criteria) belirlenmiş olması çok yararlıdır. Kapsama kriterleri, yazılım test hedeflerine ulaşılmasını sağlayan aktiviteleri yürütmek için anahtar performans göstergeleri (KPI, key performance indicator) olarak etkili bir rol oynayabilir.

Örneğin, bir mobil uygulama için, test esası gereksinimlerin bir listesini ve desteklenen mobil cihazların bir listesini içerebilir. Her gereksinim, test esasının bir unsurudur. Desteklenen her cihaz da test esasının bir unsurudur. Kapsama kriterleri(ne göre), test esasının her unsuru için en az bir test senaryosu gerektirebilir. Testler koşulduğunda(executed), bu testlerin sonuçları paydaşlara(Stakeholder) belirtilen gereksinimlerin yerine getirilip getirilmediğini ve desteklenen cihazlarda arızalar gözlenip gözlenmediğini gösterir.

(Test basis'in her test seviyesi ve test çeşidi için measurable coverage criteria(ölçülebilir kapsama kriterinin) belirlenmiş olması çok yararlıdır. Test hedeflerine ulaşılmasını sağlayan aktiviteleri yürütmek için anahtar göstergeleri(Key performance indicator) olarak etkili bir rol oynayabilir.
Her gereksinim, test esasının bir unsurudur. Kapsama kriterleri (ne göre), test esasının her unsuru için en az bir test senaryosu gerektirebilir)

1.4.2 Test Aktiviteleri ve Yapılacak İşler

Bir test süreci aşağıdaki ana aktivite gruplarından oluşur: (STLS aşamaları)

1. ***Test planlama(Test planning)***
2. ***Test gözetimi ve kontrolü(Monitoring(Testüberwachung) and control(Steurung : kumanda))***
3. ***Test analizi(Test analysis)***
4. ***Test tasarımı(Test design, Testentwurf)***
5. ***Test uyarlama(Test implementation, Testrealisierung)***
6. ***Test koşumu(Test execution, Testdurchführung)***
7. ***Test tamamlama(Test completion, Testabschluss)***

Her aktivite grubu, aşağıdaki bölümlerde açıklanacak aktivitelerden oluşur. Her aktivite grubundaki her bir aktivite, bir projeden veya sürümden(release) diğerine değişebilen birçok ayrı işi içerebilir.

Bu aktivite gruplarının birçoğu mantıksal olarak sıralı görünse de, genellikle döngüsel olarak uygulanırlar. Örneğin, çevik yazılım geliştirme(Agile development involves), devamlı planlama tarafından desteklenen ve sürekli olarak gerçekleşen yazılım tasarımı, geliştirme ve testlerin

küçük döngülerinden oluşur. Dolayısıyla, bu yazılım geliştirme yaklaşımı(Agile) içinde test aktiviteleri de döngüsel ve sürekli olarak gerçekleşir(designing, coding, testing à döngüsü).

***Sıralı yazılım geliştirmede(sequential software development)** bile, kademeli mantıksal aktivite dizisi; **kesişme(overlap), birleşme(combination), eşzamanlılık(concurrency)** veya **çıkarmayı(omission)** içerir. Bu nedenle bu ana aktivitelerin genellikle yazılım ve proje bağlamında düzenlenmesi gerekir.*

1. **Test planlama (Test planning)**

Test planlaması, test hedeflerini(amacını) belirleyen aktiviteler ve proje bağlamının dayattığı **kısıtlamalar** dâhilinde test hedeflerini yerine getirme yaklaşımını içerir (örneğin, **uygun test tekniklerini ve işlerini belirlemek ve bunları belirtilen zamanda bitirmek için bir test zaman çizelgesi oluşturmak**). Gözetim ve kontrol faaliyetlerinden gelen geri bildirimlere dayanarak test planları gözden geçirilip güncellenebilir.

(Test hedefleri, proje türünün gerektirdiği kısıtlamalar dahilinde test hedeflerini yerine getirebilmek için yapılan planlamaya denir. Yapılacak işler, hangi test tekniği kullanılacak, hangi sürede bu işler bitirilecek. Gerektiğinde güncelleme yapılabilir)

2. **Test gözetimi ve kontrolü(Test monitoring and control, Testüberwachung und -steuerung)**

Test gözetimi (Test monitoring, Testüberwachung), test planında tanımlanan test gözetim metriklerini(test monitoring metrics) kullanarak planlanan ile gerçekleşen ilerlemenin sürekli karşılaştırılmasını içerir. Test kontrolü, test planında belirlenen hedeflere ulaşmak için gerekli önlemlerin alınmasını içerir. Test gözetimi ve kontrolü, çıkış kriterlerinin(exit criteria) değerlendirilmesiyle desteklenir; bu kriterler bazı yaşam döngülerinde "Tamamlandı"(done) tanımı olarak da bahsedilir.

(Test monitoring, test planında tanımlanan test gözetim metriklerini (test monitoring metrics) kullanarak planlanan ile gerçekleşen ilerlemenin sürekli karşılaştırılmasını içerir. Test kontrolü, test planında belirlenen hedeflere ulaşmak için gerekli önlemlerin alınmasını içerir)

Örneğin, bir test seviyesi için test koşumu çıkış kriterlerinin değerlendirilmesi aşağıdakileri içerebilir:

- Test sonuçlarının(Test result) ve kayıtlarının(log, Protokoll) belirtilen kapsama kriterleriyle(coverage criteria) karşılaştırılarak **kontrol edilmesi**
- Test sonuçlarına ve kayıtlara dayanarak birim(component) veya sistemin kalite seviyesinin değerlendirilmesi
- Daha fazla test gerekip gerekmediğinin belirlenmesi (örneğin, belirli bir ürün riskinin kapsanması için ek testlerin yazılması ve koşturulması)

(Testin değerlendirmesi: testi çalıştırdıktan sonra test sonuçları ve kayıtlara(log) göre sonuçlar exit creteria yı sağlıyor mu, kalite seviyesi nasıl, daha fazla test gerekli mi değil mi diye değerlendirme yapıyoruz)

Test ilerlemesi (Test progress, Testfortschritt) , planla karşılaştırmalı olarak test ilerleme raporlarında paydaşlara(Stakeholder) iletilir; buna plandan sapmalar ve testleri durdurma kararını destekleyen bilgiler de dâhildir.

3. Test analizi(Test analysis)

Test analizi sırasında, test edilebilir özellikleri(Features) belirlemek ve ilgili test koşullarını(test conditions, Testbedingungen) tanımlamak için test esası(Testbase) analiz edilir. Diğer bir deyişle, test analizi ölçülebilir kapsama kriterleri(measurable coverage criteria) açısından "**neyin test edileceğini**" belirler.

(Test koşullarını (test condition, testbedingungen) tanımlamak ve test edilebilir özellikleri(features) belirlemek için yapılan işleme Test analiz diyoruz. "Neyin test edileceğini" belirler)

Test analizi aşağıdaki ana aktiviteleri içerir:

Test seviyesine(test level, Teststufe) uygun şekilde test esasını(test base) analiz etmek, örneğin:

- Gereksinimler, fonksiyonel gereksinimler, sistem gereksinimleri, kullanıcı hikâyeleri, epikler(Epics : içerisinde birden fazla user story'yi barındıran , 8 gün ve daha uzun süre üzerinde çalışılması gereken User Story'ler. Payment bir Epic'tir mesela), kullanım senaryoları gibi gereksinim spesifikasyonları veya istenen fonksiyonel(functional : Webelement'in davranışsal olarak bir sonuç üretebilme kabiliyeti. mesela üründen 5 tane almak için dropbox'tan beşi seçmem lazım) ve fonksiyonel olmayan(non-functional : kullanıcıya bilgi vermek için yazılmış bir text gibi) birim veya sistem davranışını belirten benzer çalışma ürünleri
- Sistem veya yazılım mimarisi şemaları veya dokümanları, tasarım spesifikasyonları, çağrı akışları, modelleme şemaları (örneğin, UML veya ER şemaları), arayüz spesifikasyonları gibi tasarım ve uygulama bilgileri veya birim veya sistem yapısını belirten benzer çalışma ürünleri
- Hayata geçirilmiş kod, veri tabanı(database), meta verileri(metadata : tanımlama bilgileri) ve sorgular(queries) ve arayüzler(interfaces)
- Birim veya sistemin fonksiyonel, fonksiyonel olmayan ve yapısal yönlerini dikkate alan **risk analizi raporları**

(epikler(Epics : içerisinde birden fazla user story'yi barındıran, 8 gün ve daha uzun süre üzerinde çalışılması gereken User Story'ler. Payment bir Epic'tir mesela)

fonksiyonel(functional : Webelement'in davranışsal olarak bir sonuç üretebilme kabiliyeti. Mesela üründen 5 tane almak için dropbox'tan beşi seçmem lazım)

Fonksiyonel olmayan(non-functional : kullanıcıya bilgi vermek için yazılmış bir text gibi))

Aşağıdakiler gibi çeşitli tiplerdeki hataları belirlemek için test esasını ve test öğelerini(test items, Testelements) değerlendirme:

- Belirsizlikler

- Çıkarmalar
- Tutarsızlıklar
- Yanlışlıklar
- Çelişkiler
- Gereksiz ifadeler

Test edilecek özelliklerin(features) ve özellik gruplarının belirlenmesi

Test esasının analizine dayanarak fonksiyonel, fonksiyonel olmayan ve yapısal özellikleri, diğer iş ve teknik faktörleri ve risk seviyelerini dikkate alarak her özellik için **test koşullarının belirlenmesi** ve **önceliklendirilmesi**

Test esasının her unsuru ve ilgili test koşulları arasında **çift yönlü izlenebilirliğin belirlenmesi** (iki yönlü takip edilebilirlik, requirement ve yazdığımız test caselerimiz var. bunların birbirleriyle bağlantıları çift yönlü takip deniyor. Hangi test için hangi requirement, hangi requirement için hangi test).

(Test analizi aktiviteleri: Test seviyesine uygun test esasının analiz edilmesi, risk analiz raporları, belirsizikler, tutarsızlıklar vs. yönüyle test esası ve test öğelerinin değerlendirilmesi, test edilecek özellikler ve özellik gruplarının belirlenmesi, test koşullarının belirlenmesi ve önceliklendirilmesi, çift yönlü izlenebilirliğin belirlenmesi.)

Kara kutu(black box), beyaz kutu(white box) ve tecrübeye dayalı test(experience-based test, erfahrungsbasierten Test) **tekniklerinin uygulanması**, test analizi sürecinde önemli **test koşullarının gözden kaçırılma olasılığını azaltmak** ve **daha kesin ve doğru test koşulları tanımlamak için** (bkz. Bölüm 4) faydalı olabilir.

(3 tane test tekniğimiz var. Black box test, White box text, experience-based test)

Bazı durumlarda, test analizi, test tüzüklerinde(Test charters) test hedefi(test objektives, Testziel) olarak kullanılacak **test koşullarını** üretir. **Test tüzükleri**, bazı tecrübeye dayalı test çeşitlerinde tipik çalışma ürünleridir (bkz. Bölüm 4.4.2). Test esasıyla bu test hedefleri ilişkilendirilebilir ise, tecrübeye dayalı testler sırasında elde edilen kapsam ölçülebilir.

(Test analizi ile test tüzüklerinde(test charters) test hedefi olarak kullanılacak test koşulları üretilir, tecrubeye dayalı test çeşitlerinde bu tipik çalışma ürünüdür. Burada test base ile test hedefleri karışlaştırılabilirse çıkan sonuçla tecrubeye dayalı testlerin başarı durumu ölçülebilir.)

Test analizi sırasında **hataların belirlenmesi,** özellikle başka bir gözden geçirme tekniğinin kullanılmadığı ve/veya test sürecinin gözden geçirme süreci (review process, Reviewprocess) ile yakından bağlantılı olmadığı durumlarda, önemli bir potansiyel faydadır.

Bu tür test analizi faaliyetleri sadece gereksinimlerin tutarlı, doğru ifade edilmiş ve eksiksiz olduğunu doğrulamakla kalmaz, aynı zamanda gereksinimlerin müşteri, kullanıcı ve diğer paydaş ihtiyaçlarını doğru şekilde karşılayıp karşılamadığının sağlamasını da yapar.

Örneğin, **kodlamadan önce** kullanıcı hikâyelerinden(**user story**) ve kabul kriterlerinden(**acceptence criteria) test koşulları(test condition)** ve **test senaryoları(test cases)** oluşturmayı içeren, davranış odaklı yazılım geliştirme (BDD, behavior driven development) ve kabul testi odaklı yazılım geliştirme (ATDD, acceptance test driven development) gibi tekniklerle kullanıcı hikâyeleri(**user story**) ve kabul kriterleri(**test conditions**) **doğrulanır, sağlaması yapılır ve hataları bulunur**.

(Kodlamadan önce davranış odaklı yazılım geliştirme (BDD, behavior driven development) ve kabul testi odaklı yazılım geliştirme (ATDD, acceptance test driven development) gibi tekniklerle user story ve test conditions doğrulanır, sağlaması yapılır ve hataları bulunur)

4. Test tasarımı(Test design, Testentwurf)

***Test tasarımı(Test design, Testentwurf)** sırasında test koşulları(test condition); **üst seviye test senaryoları(high-level test cases, abstrakte Testfälle), üst seviye test senaryo grupları(Sets aus abstrakten Testfällen)** ve test yazılımları(Testware, Testmittel) olarak detaylandırılır. Bu nedenle, test analizi "ne test edilecek?" sorusuna cevap verirken, test tasarımı "nasıl test edilecek?" sorusuna cevap verir.*

Test yazılımları(Testware): bir testi gerçekleştirebilmek için gerekli olan tüm toollar, elimizdeki Testbase'ler, Testbase'leri çalışabileceğimiz tüm dökümanlar, hepsinin oluşturduğu toplam havuza deniyor.

(Test tasarımı sırasında test conditionlarını çok detaylı olmayan test caseler ve Testware lere dönüştürülür (olarak detaylandırılır).)

Test tasarımı aşağıdaki ana aktiviteleri içerir:

Test senaryolarının ve test senaryo gruplarının tasarlanması ve önceliklendirilmesi
Test koşullarını ve test durumlarını desteklemek için gereken test verilerinin(test data, Testadetn) **belirlenmesi**
Test ortamının tasarlanması ve gerekli altyapı ve araçların belirlenmesi
Test esası, test koşulları, test senaryoları ve test prosedürleri arasında **çift yönlü izlenebilirliğin tespit edilmesi** (bkz. Bölüm 1.4.4)

Test analizinde olduğu gibi test tasarımı da test esasındaki benzer hata çeşitlerinin belirlenmesini sağlayabilir.

Yine test analizinde olduğu gibi, test tasarımı sırasında hataların belirlenmesi önemli bir potansiyel faydadır.

(Test tasarımı(Test design) aktivitileri: test case'lerin tasarlanması ve önceliklendirilmesi, test verilerinin belirlenmesi, test ortamının tasarlanması ve gerekli altyapı ve araçların belirlenmesi, test bases, test conditions, test cases ve test process'lerin arasında çift yönlü izlenebilirliğin tespit edilmesi.)

5. Test uyarlama(Test implementation, Testrealisierung)

Test uyarlama sırasında, test koşumu için eğer gerekiyorsa test yazılımı(**testware, Testmittel**) oluşturulur ve/veya tamamlanır; buna test senaryolarından test prosedürleri(test procedures) **üretmek de dâhildir**. Dolayısıyla,

test tasarımı "nasıl test edilecek?" sorusuna cevap verirken, test uyarlama(Test implementation, Testrealisierung) "şu anda testleri koşmak için gerekli şeylere sahip miyiz?" sorusuna cevap arar.(gerekli olan hazırlıkların yapılması)

Test uyarlama aşağıdaki ana aktiviteleri içerir:

Test prosedürlerinin yazılması, önceliklendirilmesi ve gerekiyorsa test betiklerinin(test scripts, Testscript: sistemin beklendiği gibi çalışıp çalışmadığını test etmek için test edilen sistemde gerçekleştirilecek bir dizi talimata denir.) **otomasyonu**

Test prosedürlerinden ve yazıldıysa otomatikleştirilmiş test betiklerinden **test grupları oluşturulması**

Test gruplarını(**Test suit**), test senaryolarını ve otomatikleştirilmiş test betiklerini(**Test scripts**) içeren test koşum çizelgesi oluşturulması(test execution schedule) (bkz. Bölüm 5.2.4).

Test ortamının(Test environment, Testumgebung) oluşturulması (gerekiyorsa test kuluçkası, servis sanallaştırması, simülatörler ve diğer altyapı öğelerini dâhil ederek) ve **ihtiyaç duyulan her şeyin doğru şekilde kurulduğunun doğrulanması**

Test verilerinin hazırlanması ve test ortamına düzgün şekilde yüklenmesinin sağlanması

Test esası, test koşulları, test senaryoları, test prosedürleri ve test grupları arasında **çift yönlü izlenebilirliğin doğrulanması ve güncellenmesi** (bkz. Bölüm 1.4.4)

(Test uyarlama aktiviteleri: Test prosedürlerinin yazılması, önceliklendirilmesi, test betiklerinin(test script) otomasyonu, test grupları oluşturulması(Test suit), Test koşum çizelgesi oluşturulması, Test ortamının(test environment, Test umgebung) oluşturulması, ihtiyaç duyulan herşeyin doğru şekilde kurulduğunun doğrulanması, Test verilerinin hazırlanması, çift yönlü izlenebilirliğin doğrulanması ve güncellenmesi.
test betiklerinin(test scripts, Testscript: sistemin beklendiği gibi çalışıp çalışmadığını test etmek için test edilen sistemde gerçekleştirilecek bir dizi talimata denir)

Test tasarımı(Test design) ve test uyarlama(Test implementation) görevleri genellikle birlikte hayata geçirilir.

Keşif testlerinde(exploratory testing, Explorativen Test) ve diğer tecrübeye dayalı test(experience-based testing, erfahrungsbasierten Tests) çeşitlerinde, test tasarımı ve uyarlaması, test koşumunun bir parçası olarak gerçekleştirilebilir ve dokümante edilebilir. Keşif testleri, test tüzüklerine(test charters) (test analizinin bir parçası olarak üretilir) dayandırılmış olabilir ve keşif testleri tasarlandıktan ve uyarlandıktan sonra hemen koşturulur (bkz. Bölüm 4.4.2).

6. Test koşumu(Test execution, Testdurchführung)

Test koşumu(Test execution, Testdurchführung) sırasında, test grupları(Test suit) test koşum çizelgesine(**test execution schedule, Testausführungplan**) uygun olarak çalıştırılır.

Test koşumu aşağıdaki ana aktiviteleri içerir:

Test öğesinin/öğelerinin(Test item, Testelement) veya test nesnesinin, test aracının/araçlarının(Testobjeckt) ve test yazılımının(Testware, Testwerkzeug) kimlik numaralarının(ID) ve versiyonlarının **kaydedilmesi**
Testlerin **manuel olarak** veya test koşum araçları(execution tools, Testausführungswerkzeugen) kullanılarak **koşulması**
Gerçekleşen sonuçlarla beklenen sonuçların karşılaştırılması
Olası nedenlerin belirlenmesi için anomalilerin analiz edilmesi (örneğin, koddaki hatalar nedeniyle arızalar oluşabilir, ancak yanlış pozitifler de ortaya çıkabilir [bakınız bölüm 1.2.3])
Gözlemlenen arızalara dayanılarak **hataların bildirilmesi** (bkz. Bölüm 5.6)
Test koşum sonucunun(outcome of test execution, Ergebnisse der Testdurchfuhrung) **kaydedilmesi** (örneğin, başarılı, başarısız, engellendi)
Bir anomali için ya da planlı testlerin bir parçası olarak **test aktivitelerinin tekrarlanması** (örneğin, düzeltilmiş bir testin, onaylama testinin(confirmation testing, Fehlernachtests) ve/veya **regresyon testlerinin** koşturulması)
Test esası, test koşulları, test senaryoları, test prosedürleri ve **test sonuçları** arasındaki çift yönlü izlenebilirliğin(bi-directional traceability, bidirektionalen Verfolgbarkeit) **doğrulanması** ve **güncellenmesi**.

(Test execution aktiviteleri: test öğelerinin(test item, test element) veya test objelerinin, test yazılımlarının ID ve versiyonların kaydedilmesi, testlerin manuel veya toollar vasıtasıyla çalıştırılması, gerçekleşen sonuçlar ile beklenen sonuçların karşılaştırılması, normal dışı durumların analiz edilmesi, olası nedenlerinin belirlenmesi, yanlış pozitifler olup olmadığına dikkat edilmesi, hataların bilidirilmesi, test sonucunun kaydedilmesi, bir anormallik olup olmadığının kontrolü için test aktivitelerinin tekrarlanması, bunun için confirmation ve regresyon testlerinin çalıştırılması, çift yönlü izlenebilirliğin doğrulanması ve güncellenmesi)

7. Test tamamlama(Test completion, Testabschluss)

Test tamamlama aktiviteleri; test projesinde elde edilen deneyimi(experience, Erfahrungen), proje boyunca üretilen test yazılımını(Testware, Testmittel) ve elde edilen diğer ilgili bilgileri(relevant information) toplar ve bir araya getirir. Test tamamlama aktiviteleri, bir yazılımın piyasaya sürülmesi(release, freigegeben), bir test projesinin tamamlanması (veya iptal edilmesi), bir çevik proje döngüsünün tamamlanması(Agile Projektiteration) (örneğin bir geriye dönük değerlendirme toplantısının parçası olarak), bir test seviyesinin(Test level, Teststufe) tamamlanması veya bir bakım sürümünün(maintenance release, Wartungsrelease) tamamlanması gibi proje kilometre taşlarında gerçekleştirilir.

(Test completion : elde edilen deneyim, test yazılımı ve gerekli tüm bilgileri toplayıp biraraya getirme faaliyetidir.

Test completion şu 5 farklı aşamalarda yapılır: yazılımın piyasaya sürülmesinde, projenin tamamlanmasında, Agile proje döngüsünün tamamlanmasında, bir test seviyesinin tamamlanmasında, bir bakım sürümünün tamamlanmasında.)

Test tamamlama aşağıdaki ana aktiviteleri içerir:

- Tüm hata raporlarının(defect reports, Fehlerbericte) **kapatılıp kapatılmadığının kontrol edilmesi**, test koşumu sonunda **çözülmeden kalan hatalar için değişiklik taleplerinin** ürün iş listesine(product backlog items, Product-Backlog-Elementen) **girilmesi**

(Product backlog: Product Backlog, üründe olması gereken tüm işlerin listelendiği, ihtiyaçları açıklayan ve öncelik sırasını gösteren tek kaynaktır. Product Backlog'un oluşturulmasından, değiştirilmesinden ve sıralanmasından Product Owner sorumludur.

Product backlog-items: Product Backlog'a kaydedilen her bir kaleme "Product Backlog Item (PBI) veya Türkçe adı ile "Ürün Gereksinim Maddesi" denir)

Paydaşlara iletilmek üzere bir test özet raporunun(test summary report, Testabschlussberichts) **oluşturulması**

Test ortamının(test invorenment, Testumgebung), test verilerinin, test altyapısının(test infrastructure, Testinfrastruktur) ve diğer test yazılımlarının daha sonra tekrar kullanılmak üzere sonlandırılması ve arşivlenmesi

Test yazılımının bakım ekiplerine(maintenance teams, Wartungsteams), diğer **proje ekiplerine** ve/veya test yazılımının kullanımından **faydalanabilecek diğer paydaşlara teslim edilmesi**

Gelecek döngüler(yineleme), sürümler ve projeler için **gereken değişikliklerin belirlenmesi amacıyla** tamamlanan test faaliyetlerinden **çıkarılan derslerin analiz edilmesi**

Test süreci olgunluğunun artırılması için toplanan bilgilerin kullanılması

(Test tamamlama aktiviteleri: hata raporlarının kapatılıp kapatılmıdığının kontrol edilmesi. Çözülmeden kalan hatalar için değişiklik taleplerinin ürün iş listesine girilmesi(product backlog). Stakeholderlar için test raporu oluşturulması. Test ortamı, test verileri ve diğer test yazılımlarının sonlandırılması ve arşivlenmesi. Test yazılımının bakım ekiplerine, proje ekiplerine ve faydalanabilecek stakeholderlara teslim edilmesi. Gelecek yinelemeler,sürümler ve projeler için gereken değişikliklerin belirlenmesi amacıyla çıkarılan derslerin analiz edilmesi. Test süreci olgunluğunun artırılması için toplanan bilgilerin kullanılması.)

1.4.2 Test Çalışma Ürünleri(Test work products, Testarbeitsergebnisse)

Test çalışma ürünleri(Test work products, Testarbeitsergebnisse), test sürecinin bir parçası olarak oluşturulur. Kurumların test sürecini uygulama şekillerinde önemli farklılıklar olduğu gibi, bu süreçte oluşturulan çalışma ürünleri çeşitlerinde, bu çalışma ürünlerinin düzenlenme ve yönetilme şekillerinde ve bu çalışma ürünleri için kullanılan isimlerde de önemli farklılıklar olabilir.

Test planlama çalışma ürünleri

Test planlama çalışma ürünleri genellikle **bir veya daha fazla test planı içerir**. Test planı, diğer test çalışma ürünlerinin izlenebilirlik bilgileri sayesinde ilişkili olacağı **test esası hakkında bilgiler** (aşağıya ve bölüm 1.4.4'e bakınız) ve test gözetimi ve kontrolü sırasında kullanılacak **çıkış kriterlerini** (veya Tamamlandı tanımını) **içerir.**

(Test planing work product: Test planı, ilgili test sürecinin test base'i hakkında bilgiler ve exit criteria ürünlerinden oluşur.)

Test gözetimi ve kontrolü çalışma ürünleri(Test monitoring and control work products, Arbeitsergebnisse der Testplanung)

Test gözetimi ve kontrolü çalışma ürünleri genellikle, test ilerleme raporları (sürekli ve/veya düzenli olarak üretilir) ve test özet raporları (çeşitli kilometre taşlarında üretilir) dâhil olmak üzere birçok test raporu çeşitlerini içerir.

Tüm test raporları, test koşum sonuçlarının özetlenmesi de dâhil olmak üzere, rapor tarihi itibariyle test ilerlemesi hakkında ilgili paydaşları ilgilendiren ayrıntıları sağlamalıdır.
Test gözetimi ve kontrolü çalışma ürünleri ayrıca **görev tamamlama**, **kaynak tahsisi ve kullanımı** ve **efor** gibi proje yönetimi konularını da ele almalıdır.

(Test monitoring ve kontrol work products: test ilerleme raporları, test özet raporları, test execut sonuçları, rapor tarihi itibariyle test ilerlemesi, görevlerin tamamlanma durumları, kaynak tahsisi, kullanımı ve efor gibi stakeholderları ilgilendiren ayrıntıların raporlarından oluşur.)

Test analizi çalışma ürünleri(Test analysis work products, Arbeitsergebnisse der Testanalyse)

Test analizi çalışma ürünleri, **tanımlanmış** ve **önceliklendirilmiş test koşullarını** içerir; bu koşulların her biri idealde kapsadığı test esasının unsurlarına **çift yönlü olarak izlenebilir** olmalıdır. Keşif testleri(exploratory testing, exploratives Testen) için test analizi, **test tüzüğünün(test charters) oluşturulmasını** içerebilir. Test analizi ayrıca, test esasındaki **hataların belirlenmesi ve raporlanmasıyla** sonuçlanabilir.

(Test analysis work product: çift yönlü izlenebilir olarak tanımlanmış ve önceliklendirilmiş test conditions(Testbedingungen), test tüzüğünün oluşturulması(test charters), hataların tespit ve raporlanması gibi ürünlerden oluşur)

Test tasarımı çalışma ürünleri(Test design work products, Arbeitsergebnisse des Testentwurfs)

Test tasarımı, test analizinde belirlenen test koşullarını uygulamak için **test senaryoları ve test senaryosu gruplarının belirlenmesi** ile sonuçlanır. Girdi verileri ve beklenen sonuçlar için somut değerlere atıfta bulunmadan, **üst seviye test senaryoları tasarlamak** genellikle iyi bir uygulamadır. Bu üst seviye test senaryoları, farklı somut verilere sahip birçok test döngüsü boyunca tekrar kullanılabilir ve test senaryosunun kapsamını yeterli ölçüde dokümante eder. İdeal olarak, her test senaryosu, kapsadığı test koşuluna/koşullarına **çift yönlü olarak izlenebilirdir.**

Test tasarımı aynı zamanda **gerekli test verilerinin** tasarlanması ve/veya **belirlenmesi,** test ortamının **(test invorenment, Testumgebung) tasarımı ve altyapı ve araçların belirlenmesi** ile sonuçlanır, ancak bu sonuçların ne ölçüde dokümante edildiği projeye göre büyük miktarda farklılık gösterir.

Test analizinde belirlenen test koşulları, test tasarımında daha da geliştirilebilir.

(Test design work product: Gerekli test verilerinin belirlenmesi. Test ortamının tasarımı ve araçlarının belirlenmesi. Çift yönlü izlenebilir şekilde High-level olarak farklı somut datalarla Test case ve Test case gruplarının belirlenmesi. Test caselerin oluşturulması)

Test uyarlama çalışma ürünleri(Test implementation work products, Arbeitsergebnisse der Testrealisierung)

Test uyarlama çalışma ürünleri aşağıdakileri içerir:

Test prosedürleri ve bu test prosedürlerinin sıralanması
Test grupları
Test koşum çizelgesi

İdeal olarak, test uyarlaması tamamlandıktan sonra, test planında belirlenen kapsama kriterlerinin karşılandığı, test senaryoları ve test koşulları kullanılarak test prosedürleri(Test procedures, Testkonzept) ve test esasının belirli unsurları arasındaki çift yönlü izlenebilirlik ile kanıtlanabilir.

Bazı durumlarda, **test uyarlamada**, servis sanallaştırma ve otomatize t**est betikleri(test scripts) gibi araçları kullanan** veya onlar tarafından kullanılan çalışma ürünleri oluşturulur.

Test uyarlama ayrıca **test verilerinin ve test ortamının oluşturulması ve doğrulanmasını içerebilir**. Veri ve/veya ortam doğrulama sonuçlarının dokümantasyonu önemli ölçüde farklılık gösterebilir.

Test verileri, test senaryolarının girdilerine ve beklenen sonuçlarına somut değerler atamakta kullanılır. Bu gibi somut değerler, somut değerlerin kullanımıyla ilgili açık talimatlarla birlikte, üst seviye test senaryolarını uygulanabilir alt seviye test senaryolarına dönüştürür. Aynı üst seviye test senaryosu, test nesnesinin farklı sürümlerinde çalıştırıldığında farklı test verileri kullanabilir. Somut test verileriyle ilişkili beklenen somut sonuçlar, bir test oracle'ı kullanılarak belirlenir.

Keşif testlerinde(exploratory testing, exploratives Testen), test koşumu sırasında bazı **test tasarım ve uyarlama çalışma ürünleri oluşturulabilir**, ancak keşif testlerinin (ve bu testlerin test esasındaki belirli öğelere izlenebilirliklerinin) dokümante edilme dereceleri önemli ölçüde değişebilir.

Test analizinde belirlenen test koşulları, test uyarlamada daha da geliştirilebilir.
(Test implementation work product : gerekli olan hazırlıkların tamamlandığına dair gerekli olan bilgiler.)

Test koşumu çalışma ürünleri

Test koşumu çalışma ürünleri şunları içerir:

- Bireysel test senaryolarının veya **test prosedürlerinin durumunun dokümantasyonu** (örn. Çalışmaya hazır, geçme, başarısız olma, bloke, kasıtlı olarak atlama vb.)
- **Hata raporları** (bkz. Bölüm 5.6)
- Teste **hangi test öğesi** (leri), **test nesnesi** (leri), **test araçları** ve **test yazılımının** dahil edildiğine ilişkin belgeler İdeal olarak, **test yürütme tamamlandıktan sonra**, test esasının her bir öğesinin durumu belirlenebilir ve iki - ilişkili test prosedür (ler) i ile çift yönlü izlenebilirlik. Örneğin, hangi gereksinimlerin planlanan tüm testleri geçtiğini, hangi gereksinimlerin başarısız olduğunu ve / veya bunlarla ilişkili kusurların olduğunu ve hangi gereksinimlerin hala çalıştırılmayı bekleyen planlanmış testlere sahip olduğunu söyleyebiliriz. Bu Kapsam kriterlerinin karşılandığının doğrulanmasını sağlar ve test sonuçlarının paydaşların anlayabileceği şekilde raporlanmasını sağlar.

(Test execute work product : Test prosedürlerinin durumunun dökümantasyonu, hata raporları, bi-direct tracebility(çift yönlü doğrulama) ile test execute edildikten sonra test öğeleri, test objeleri, test araçları ve testware(Testmittel)'lerin her birinin gereksinimleri geçip geçmediğinin raporu.)

Test tamamlama çalışma ürünleri

Test tamamlama çalışma ürünleri, **test özet raporlarını(test summary report, Testabschlussberichte)**, sonraki projelerin veya yinelemelerin iyileştirilmesi için eylem öğelerini, değişiklik taleplerini veya ürün iş listesi öğelerini(**Product backlog items, Product Backlog-Elemente**) ve sonlandırılmış test yazılımını içerir.

(Test completion work product: Test özet raporları, sonraki projelerin iyileştirilmesi adına teklifler, product backlog-items)

1.4.3 Test Esası ve Test Çalışma Ürünleri Arasında İzlenebilirlik

Bölüm 1.4.3'te belirtildiği gibi, test çalışma ürünleri ve bu çalışma ürünlerinin isimleri önemli ölçüde değişmektedir. Bu değişikliklerden bağımsız olarak, etkin test gözetimi ve kontrolünü uygulamak için, yukarıda anlatıldığı gibi, **test süreci boyunca test esasının(test base) her unsuru ile bu unsurla ilişkili**

çeşitli test çalışma ürünleri arasında izlenebilirliğin(traceability, verfolgbarkeit) oluşturulması ve sürdürülmesi önemlidir.

(Traceability nin amacı bir noktaya geldikten sonra geriye dönüp, yapılan işler gereksinimleri karşılıyor mu, dönüp bunlara bakabilmek, bunun kontrolünü yapmak.

Jira da elimizdeki requirementlere göre Test case lerimizi yazıyoruz, hangi requirement hangi Test case için yazdığımızın belli olması lazım ki takip edilmeyen hiç bir bölüm kalmasın. execut edildiğinde buna göre varsa bug report yazılması lazım. Bu takibi yapmak için kullandığımız sisteme Traceability deniyor. Bu eskiden excelle yapılıyormuş, bunlar günümüzde projekt management dediğimiz tool lar kullanılıyor.Jira da bu x-ray ile birlikte kullanılarak icra ediliyor.)

Test kapsamının değerlendirilmesine ek olarak, iyi bir izlenebilirlik aşağıdakileri de destekler:

- Değişikliklerin etkisinin analiz edilmesi
- Testlerin denetlenebilir şekilde yapılması
- IT (BT) yönetim kriterlerinin(IT governance criteria) karşılanması
- Test esasındaki unsurların durumunu (ör. testleri geçen gereksinimler, testlerde başarısız olan gereksinimler ve bekleyen testleri olan gereksinimler) içerecek şekilde **test ilerleme raporlarının(test progress reports, Testfortschrittsberichten)** ve **test özet raporlarının(test summary reports, Testabschlussberichten) anlaşılabilirliğinin artırılması**
- **Testlerin** teknik yönlerinin(technical aspects, technischen Aspekte), **anlaşılır şekilde paydaşlara aktarılması**
- İş hedeflerine göre **ürün kalitesinin**, süreç kapasitesinin(process capability, Prozessfähigkeit) ve **proje ilerlemesinin değerlendirilmesi için bilgi sağlanması**

(iyi bir izlenebilirlikte; değişiklikler analiz edilebilmeli, testler denetlenebilir olmalı, test ilerleme ve özet raporları anlaşılabilir olmalı, ürün kalitesi ve süreç ilerlemesi hakkında bilgi sağlamanlı)
duydukları bilgi izlenebilirliğini sağlamak için kendi yönetim sistemlerini geliştirirler.

1.5 Test Etme Psikolojisi

1.5.1 İnsan Psikolojisi ve Test

Gereksinimlerin gözden geçirilmesi veya kullanıcı hikâyesini olgunlaştırma oturumu gibi statik bir test sırasında hataların bulunması veya dinamik test koşumu sırasında arızaların bulunması, yazılım ürününün ve yazılım ürününü geliştirenin eleştirilmesi olarak algılanabilir. **Psikolojide yer alan "doğrulama sapması" tezine göre, mevcut görüş ve inançlara uymayan bilgilerin kabul edilmesi kolay değildir.** Örneğin, yazılımcılar kodlarının hatasız olmasını beklediklerinden, kodun hatalı olduğunu kabul etmelerini zorlaştıran bir doğrulama sapmasına sahiptirler. Doğrulama sapmasının yanında diğer bilişsel eğilimler insanların testlerde elde edilen bilgileri anlamalarını veya kabul etmelerini zorlaştırabilir. Ayrıca, kötü haber elçilerini suçlamak ortak bir insani özelliktir ve testlerde üretilen bilgiler sıklıkla kötü haberler içerir.

(Psikolojide yer alan "doğrulama sapması" tezine göre, mevcut görüş ve inançlara uymayan bilgilerin kabul edilmesi kolay değildir.
Yazılımcılar kodlarının hatasız olmasını beklediklerinden, kodun hatalı olduğunu kabul etmelerini zorlaştıran bir doğrulama sapmasına sahiptirler)

Bu psikolojik faktörlerin bir sonucu olarak, projenin ilerlemesine ve ürün kalitesine büyük katkı sağlasa da, bazı insanlar testleri yıkıcı bir faaliyet olarak algılayabilir (bkz. Bölüm 1.1 ve 1.2). Bu algıları azaltmak için, hatalar ve arızalar hakkındaki bilgiler yapıcı bir şekilde iletilmelidir. Bu şekilde, test uzmanları ile analistler, ürün sahipleri, tasarımcılar ve yazılımcılar arasındaki gerilimler azaltılabilir. Bu hem statik hem de dinamik testler için geçerlidir.

Test uzmanları ve test yöneticilerinin, hataları, arızaları, test sonuçlarını, test ilerlemesini ve risklerini sağlıklı bir şekilde paylaşabilmesi ve meslektaşları ile olumlu ilişkiler kurabilmesi için iyi insani ilişkiler kurma becerilerine sahip olması gerekir. İyi iletişim kurmanın yollarına aşağıdakiler örnek verilebilir:

- Savaşarak değil iş birliği yaparak başlayın. Herkese daha iyi kalitede yazılımlar geliştirmek olan ortak amacınızı hatırlatın.
- Testin faydalarını vurgulayın. Örneğin, ürün sahiplerine onlarla paylaşacağınız hata bilgilerinin, kendi çalışma ürünlerini ve becerilerini geliştirmelerinde yardımcı olabileceğini anlatın. Kurum genelinde, testler sırasında bulunan ve çözülen hataların, zamandan ve paradan tasarruf sağlayacağını ve ürün kalitesindeki genel riski azaltacağını anlatın.
- Hatalı öğeyi oluşturan kişiyi eleştirmeden, test sonuçlarını ve diğer bulguları tarafsız ve gerçekçi bir şekilde iletin. Objektif ve gerçeklere dayalı hata raporları yazın ve bulguları gözden geçirin.
- Karşınızdaki kişinin nasıl hissettiğini ve verdiğiniz bilgilere olumsuz tepki vermesinin sebeplerini anlamaya çalışın.
- Karşınızdaki kişinin söylediğiniz şeyi anladığını ve kendinizin de onun söylediği şeyi anladığınızı doğrulayın.

1.5.2 Test Uzmanlarının ve Yazılımcıların Düşünce Tarzları

Yazılımcılar ve test uzmanları genellikle farklı düşünür. Yazılım geliştirmenin temel amacı bir ürün tasarlamak ve üretmektir. Daha önce tartışıldığı gibi, testlerin hedefleri arasında ürünün doğrulanması ve sağlamasının yapılması, canlıya alınmadan önce hataların bulunması ve benzeri işler yer alır. Bunlar farklı düşünce tarzları gerektiren farklı işlerdir. Bu düşünce tarzlarını bir araya getirmek ürün kalitesinin artmasına yardımcı olur.

Bir düşünce tarzı, karar verme ve problem çözme için bir bireyin varsayımlarını ve tercih ettiği yöntemleri yansıtır. **Bir test uzmanının düşünce tarzı merak, profesyonel karamsarlık, eleştirel bir bakış açısı, ayrıntılara dikkat, iyi ve pozitif iletişim ve ilişkiler için motivasyon içermelidir.** Bir test uzmanının düşünce tarzı, test uzmanı deneyim kazandıkça gelişme ve olgunlaşma eğilimindedir.

(Bir test uzmanının düşünce tarzı merak, profesyonel karamsarlık, eleştirel bir bakış açısı, ayrıntılara dikkat, iyi ve pozitif iletişim ve ilişkiler için motivasyon içermelidir)

Bir yazılımcının düşünce tarzı, bir test uzmanının düşünce tarzının bazı unsurlarını içerebilir, ancak yazılımcılar, genellikle çözümler tasarlamak ve oluşturmakla ilgilenir, bu çözümlerde neyin yanlış olabileceğini fazla düşünmez. Ek olarak, psikolojideki doğrulama sapması da kendi çalışmalarındaki hataları bulmalarını zorlaştırır.

Doğru düşünce tarzı ile yazılımcılar kendi kodlarını test edebilir. Farklı yazılım geliştirme yaşam döngüsü modellerinde genellikle test uzmanları ve test aktivitelerini organize etmenin yolları da farklıdır. Test aktivitelerinin bazılarının bağımsız test uzmanları tarafından yapılması, hata bulma etkinliğini arttırır; bu da büyük, karmaşık veya hayati önem taşıyan yazılımlar için özellikle önemlidir. Bağımsız test uzmanları, yazılımı geliştirenlerden farklı bilişsel eğilimlere sahip olduklarından, ürüne ürün sahiplerinden (örneğin iş analistleri, ürün sahipleri, tasarımcılar ve programcılar) farklı bir bakış açısı getirir.

CHAPTER 2

2. Yazılım Geliştirme Yaşam Döngüsü Boyunca Test

Anahtar kelimeler

kabul testi, alfa testi, beta testi, ticari paket yazılım (COTS), birim entegrasyon testi, birim testi, onaylama testi, sözleşmeye dayalı kabul testi, fonksiyonel test, etki analizi, entegrasyon testi, bakım testi, fonksiyonel olmayan test, operasyonel kabul testi, regresyon testi, yasal düzenlemeye dayalı kabul testi, sıralı geliştirme modeli, sistem entegrasyon testi, sistem testi, test esası, test senaryosu, test ortamı, test seviyesi, test nesnesi, test hedefi, test çeşidi, kullanıcı kabul testi, beyaz kutu testi

Keywords
acceptance testing, alpha testing, beta testing, change-related testing, commercial off-the-shelf (COTS), component integration testing, component testing, confirmation testing, contractual acceptance testing, functional testing, impact analysis, integration testing, maintenance testing, non-functional testing, operational acceptance testing, regression testing, regulatory acceptance testing, sequential development model, system integration testing, system testing, test basis, test case, test environment, test level, test object, test objective, test type, user acceptance testing, white-box testing

Schlüsselbegriffe
Abnahmetest, änderungsbezogenes Testen, Alpha-Test, Auswirkungsanalyse, Benutzerabnahmetest, Beta-Test, betrieblicher Abnahmetest, Fehlernachtest, funktionaler Test, Integrationstest, kommerzielle Standardsoftware, Komponentenintegrationstest, Komponententest, nicht-funktionaler Test, Regressionstest,regulatorischer Abnahmetest, sequenzielles Entwicklungsmodell, Systemintegrationstest, Systemtest, Testart, Testbasis, Testfall, Testobjekt, Teststufe, Testumgebung, Testziel, vertraglicher Abnahmetest, Wartungstest, White-Box-Test

1.1 Yazılım Geliştirme Yaşam Döngüsü Modelleri

2.1.1 Yazılım Geliştirme ve Yazılım Testi

Uygun test aktivitelerinin gerçekleştirilebilmesi için yaygın kullanılan **yazılım geliştirme yaşam döngüsü modellerine aşina olmak, test uzmanı rolünün önemli bir parçasıdır.**

Her yazılım geliştirme yaşam döngüsü modelinde, iyi bir test pratiğinin birkaç özelliği vardır:

- Her **yazılım geliştirme aktivitesine** karşılık gelen **bir test aktivitesi** vardır.
- Her **test seviyesinin bu seviyeye özgü test hedefleri** vardır.
- Belirli bir test seviyesi için **test analizi ve tasarımı**, **ilgili yazılım geliştirme faaliyeti sırasında başlar.**

- **Test uzmanları**, gereksinimleri ve tasarımı belirlemek ve iyileştirmek için tartışmalara katılır ve **taslaklar hazırlanır hazırlanmaz çalışma ürünlerinin** (örneğin; gereksinimler, tasarım, kullanıcı hikâyeleri vb.) **gözden geçirilmesinde yer alır.**

Hangi yazılım geliştirme yaşam döngüsü modeli seçilirse seçilsin, **test faaliyetleri yaşam döngüsünün erken aşamalarında başlamalı ve testleri erkenden yapma ilkesine uyulmalıdır.**

Bu ders programı yaygın yazılım geliştirme yaşam döngüsü modellerini aşağıdaki sınıflara ayırır:

- Sıralı yazılım geliştirme modelleri(Sequential development models, Sequenzielle Entwicklungsmodelle)
- Döngüsel ve artımlı yazılım geliştirme modelleri(Iterative and incremental development models Iterative und inkrementelle Entwicklungsmodelle)

Sıralı bir yazılım geliştirme modeli, yazılım geliştirme sürecini doğrusal, **sıralı bir faaliyetler akışı** olarak tanımlar. Bu, yazılım geliştirme sürecinde **her aşamanın önceki aşama tamamlandığında başlaması gerektiği** anlamına gelir. Teorik olarak aşamaların kesişimi yoktur, ancak **uygulamada takip eden aşamadan erken geri bildirim almak faydalıdır.**
(Sıralı yazılım geliştirme modellerinde(Sequential development models) faaliyetler sıralıdır, her aşama bir önceki aşama tamamlandıktan sonra başlar.)

Şelale modelinde(Waterfall model, Wasserfallmodell), **yazılım geliştirme faaliyetleri** (örneğin; gereksinim analizi, tasarım, kodlama, testler) **birbiri ardına tamamlanır.**

Bu modelde(Waterfall), test aktiviteleri ancak diğer tüm yazılım geliştirme aktiviteleri tamamlandıktan sonra gerçekleştirilir.(teoriik olarak sıralı bir işleyiş olduğundan aşamalarda kesişmeler yaşanmaz.)

V-modeli**, test sürecini yazılım geliştirme sürecinin tamamına entegre ederek erken test prensibini uygular.** Ayrıca, V-modeli, karşılık gelen **her** yazılım geliştirme **aşaması ile ilgili test seviyeleri içerir**; bu da **erken testi desteklemektedir** (test seviyelerinin açıklaması için bölüm 2.2'ye bakınız). Bu modelde, **her test seviyesiyle ilişkili testlerin yürütülmesi sıralı olarak ilerler**, ancak bazı durumlarda kesişmeler gerçekleşir.
(V-modeli sıralı değildir, aynı anda farklı aşamalarda testler ilerleyebilir, test seviyesine göre sıralı olarak testler yürütülür. Bu model erken test prensibini uygular.)

Sıralı yazılım geliştirme modelleri tüm özellikleri tamamlanmış bir yazılım oluşturulduktan sonra yazılımın paydaşlar ve kullanıcılarla paylaşılmasını hedefler, bu yüzden de yazılımın canlıya(release) alınması için aylar veya yıllar gerekebilir.

Artımlı yazılım geliştirme(Incremental development, inkrementellen Entwicklung) modelleri ise, **parçalar halinde gereksinimleri belirlemeyi, sistemi tasarlamayı, oluşturmayı ve test etmeyi içerir;** bu, **yazılımın özelliklerinin adım adım artması anlamına gelir.** Bu artışlarda ele alınacak **yazılım özelliklerinin boyutu değişiklik gösterebilir**, bazı modellerde büyük parçalar, bazılarında ise daha küçük parçalar bulunur. Ele alınacak bir yazılım özelliği, yeni bir sorgu seçeneği veya kullanıcı arayüzü ekranında yapılan tek bir değişiklik kadar küçük olabilir.

(parçalar halinde gereksinimleri belirlemeyi sistemi tasarlamayı, oluşturmayı ve test etmeyi içerir. Bu haliyle yazılım özellikleri adım adım artarak büyür.)

Döngüsel yazılım geliştirme(Iterative development, Iterative Entwicklung), ele alınan yazılım özelliklerinin genellikle **sabit bir süreye sahip bir dizi döngüde belirlenmesi, tasarlanması, oluşturulması ve birlikte test edilmesi anlamına gelir.** Döngüler, daha önceki döngülerde geliştirilen yazılım özelliklerindeki değişikliklerin yanı sıra proje kapsamındaki değişiklikleri de içerebilir. **Son yazılım teslim edilinceye veya yazılım geliştirme durdurulana kadar her döngü, genel yazılım özellikleri kümesinin büyüyen bir alt kümesi olan çalışan yazılımı sunar.**

(sabit bir süreye sahip bir dizi döngüde belirlenmesi, tasarlanması, oluşturulması ve birlikte test edilmesi anlamına gelir. Son yazılım teslim edilinceye veya yazılım geliştirme durdurulana kadar her döngü, genel yazılım özellikleri kümesinin büyüyen bir alt kümesi olan çalışan yazılımı sunar)

Döngüsel yazılım geliştirme(Iterative development) bazı örnekleri şunlardır:

- Rational Unified Process: Her döngü göreceli olarak uzun olma eğilimindedir (örneğin iki ila üç ay) ve döngülerde ele alınan yazılım özellikleri ilgili yazılım özelliklerini de içine alacak şekilde iki veya üç yazılım özelliği grubunu içerir.
- Scrum: Her döngü göreceli olarak kısa olma eğilimindedir (örneğin, birkaç saat, gün veya hafta) ve ele alınan yazılım özellikleri, birkaç geliştirme ve/veya iki veya üç yeni yazılım özelliği gibi, aynı şekilde küçüktür.
- Kanban: Sabit uzunluklu döngüler olmadan uygulanır; tamamlandıktan sonra tek bir geliştirme veya yazılım özelliği sunabilir veya bir kerede sürüm için gerekli yazılım özelliklerini bir grup halinde sunabilir.
- Spiral (veya prototipleme): Görsel veya deneysel yazılım özellikleri oluşturmayı içerir, bunlardan bazıları ilerleyen aşamalarda büyük oranda güncellenebilir veya bunlardan vazgeçilebilir.

Bu yöntemler kullanılarak geliştirilen yazılımlar, genellikle geliştirme boyunca **test seviyeleri ile kesişir** ve **bu seviyelerin tekrar tekrar uygulanmasını içerirler**. İdeal olarak, her yazılım özelliği canlıya alıma doğru ilerlerken **çeşitli test seviyelerinde test edilir**. Bazı durumlarda, ekipler sürekli teslimat(continuous delivery, kontinuierliche Auslieferung) veya sürekli canlıya alma(continuous deployment , kontinuierliche Bereitstellung) yaklaşımlarını kullanır; bunların her ikisi de teslimat hatlarının bir parçası olarak **birçok test seviyesinin önemli miktarda otomasyonunu içerir**. Bu yöntemlerin kullanıldığı birçok yazılım geliştirme çalışması **kendi kendini yöneten ekipler kavramı**nı da içerir; bu da test çalışmasının düzenlenme şeklini ve test uzmanları ve yazılımcılar arasındaki ilişkiyi değiştirebilir.

Bu yöntemler, büyüyen bir sistem oluşturur; bu yöntemlerde canlıya alma yazılım özelliği bazında, döngü bazında veya daha klasik bir ana sürüm bazında olabilir. Yazılım özelliklerinin canlıya alınıp alınmadığına bakılmaksızın, yazılım büyüdükçe regresyon testleri giderek önem kazanır.

Sıralı modellerin(Sequential development models) aksine, döngüsel(Iterative development) ve artımlı modeller(Incremental development) haftalar veya hatta günler içinde kullanılabilir yazılımlar sunabilir, ancak gereksinimlerin tamamını içeren tam bir ürün setinin sunulması aylar, hatta yıllar sürebilir.

2.1.2 Proje Bağlamında Yazılım Geliştirme Yaşam Döngüsü Modelleri

Yazılım geliştirme yaşam döngüsü modelleri, proje bağlamına ve ürün özelliklerine göre seçilmeli ve uyarlanmalıdır. Proje hedefine, geliştirilen ürün çeşidine, kurum önceliklerine (örneğin, pazara sürülme süresi) ve belirlenmiş ürün ve proje risklerine göre uygun bir yazılım geliştirme yaşam döngüsü modeli seçilmeli ve uyarlanmalıdır. Örneğin, küçük bir dâhili idari yönetim yazılımının geliştirilmesi ve test edilmesi, bir otomobilin fren kontrol sistemi gibi emniyet açısından kritik bir sistemin geliştirilmesi ve test edilmesinden farklı olmalıdır. Başka bir örnek olarak, kurum içindeki organizasyonel ve kültürel sorunlar, ekip üyeleri arasındaki iletişimi engelleyerek yazılım geliştirme için o kurumda döngüsel yazılım geliştirme modelinin seçilmesini anlamsız hale getirebilir.

Projenin bağlamına bağlı olarak, **test seviyelerinin** ve/veya t**est aktivitelerinin birleştirilmesi veya yeniden düzenlenmesi gerekebilir.** Örneğin, ticari paket yazılımın (COTS) daha büyük bir sisteme entegrasyonu için, **birlikte çalışabilirlik testleri sistem entegrasyonu testi seviyesinde** (örneğin; altyapıya ve diğer sistemlere entegrasyon) ve **kabul testi seviyesinde** (kullanıcı kabul testleri ve operasyonel kabul testleri ile birlikte fonksiyonel ve fonksiyonel olmayan testler) yapılabilir.

Ek olarak, yazılım geliştirme yaşam döngüsü modellerinin kendileri de birleştirilebilir. Örneğin, front-end kullanıcı arayüzünü (UI) ve fonksiyonalitesini geliştirmek ve test etmek için çevik yazılım geliştirme modeli kullanılabilirken, backend sistemlerinin geliştirilmesi ve test edilmesi ve entegrasyonu için V-modeli kullanılabilir. Bir projenin erken dönemlerinde prototipleme kullanılabilir ve deneysel faz tamamlandıktan sonra artımlı geliştirme modeli(**incremental development model)** benimsenebilir.

Aygıtlar, ürünler ve hizmetler gibi birçok farklı nesneden oluşan Nesnelerin İnterneti (IoT) sistemlerinin geliştirilmesinde, genellikle her nesne için ayrı yazılım geliştirme yaşam döngüsü modelleri uygulanır. Bu durum, Nesnelerin İnterneti sisteminde yeni versiyonların geliştirilmesinde ayrı bir zorluk oluşturur. Ayrıca, bu tip sistemlerin yazılım geliştirme yaşam döngüsünde, canlıdaki kullanım (örneğin; çalıştırma, güncelleme ve kullanım dışı bırakma aşamaları) yazılım geliştirme yaşam döngüsünün ilerleyen aşamalarını daha fazla etkiler.

Yazılım geliştirme modellerinin proje bağlamına uyarlanması gerektiğinin nedenleri ve ürün özellikleri şunlar olabilir:

- Sistemlerin ürün risklerindeki farklılık (karmaşık veya basit proje)
- Birçok iş birimi bir projenin veya programın parçası olabilir (sıralı ve çevik geliştirmenin birleşimi)
- Pazara ürün sunmak için kısa süre (test seviyelerinin birleştirilmesi ve / veya test türlerinin test seviyelerine entegrasyonu)

(Proje bağlamına göre yazılım geliştirme modeli uyarlanmasının nedenleri : Sistemlerin ürün risklerindeki farklılık, projede birçok işbirimi bulunup her biri ayrı bir yazılım geliştirme modeli kullanabilir bu gibi durumlarda bu modellerin birleştirilerek uyarlanması gerekir, pazara ürün sunmak için kısa sürenin gerekli olduğu durumlarda)

2.2 Test Seviyeleri(Test levels, Teststufen)

Test seviyeleri(Test levels, Teststufen), birlikte düzenlenen ve yönetilen test aktivite gruplarıdır. **Her test seviyesi**, bölüm 1.4'te açıklanan, belirli bir yazılım geliştirme seviyesindeki yazılımla ilgili olarak gerçekleştirilen, bağımsız birimlerden veya bileşenlerden tam sistemlere veya bazı durumlarda sistemlerin sistemlerine kadar **farklı boyutlardaki yazılımları içerir**. **Test seviyeleri**, **yazılım geliştirme yaşam döngüsü içindeki diğer aktivitelerle bağlantılıdır.**

Bu ders programında kullanılan **test seviyeleri** aşağıda verilmiştir:

- ***Birim testleri(Component testing)***
- ***Entegrasyon testleri(Integration testing)***
- ***Sistem testleri(System testing)***
- ***Kabul testleri(Acceptance testing)***

Test seviyelerinin karakteristik özellikleri aşağıda verilmiştir:

- Spesifik hedefler(Specific objectives)
- Test senaryolarını oluştururken referans alınan test esası
- Test nesnesi (test edilen şey)
- Tipik hatalar ve arızalar
- Spesifik yaklaşımlar ve sorumluluklar

(Test seviyesine göre, spesifik hedefler, test esasları(test base), test nesneleri(test obje), hatalar(defects) ve arızalar(failures), yaklaşımlar ve sorumluluklar değişir. Her bir test levelının bunlara göre değişen karakteristik özellikleri vardır. Her test seviyesi için uygun bir test ortamı(environment, Umgebung) gereklidir.)

Her test seviyesi için uygun bir test ortamı(environment, Umgebung) gereklidir. Örneğin kabul testinde(acceptance testing, Abnahmetest) canlı ortam benzeri bir test ortamı ideal iken, birim testinde yazılımcılar genellikle kendi yazılım geliştirme ortamlarını kullanırlar.

2.2.1 Birim Testi(Component Testing)

Birim testinin hedefleri

Birim testleri(Component Testing) (bileşen(unit) veya modül(modül) testleri olarak da bilinir), ayrı olarak test edilebilen bağımsız birimlere odaklanır.

Birim testinin hedefleri aşağıda verilmiştir:

- Riskin azaltılması

- Birimin **fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması**
- Birimin kalitesine dair güven oluşturulması
- Birimdeki hataların bulunması
- Hataların gözden kaçarak daha üst test seviyelerine girmesinin önlenmesi

Bazı durumlarda, özellikle kod değişikliklerinin sürekli olarak devam ettiği artımlı ve döngüsel geliştirme modellerinde (örneğin çevik), otomatik birim regresyon testleri, değişikliklerin mevcut birimleri bozmadığına dair güven oluşturulmasında önemli bir rol oynar.

Birim testleri genellikle yazılım geliştirme yaşam döngüsü modeline ve sisteme bağlı olarak, sahte (mock) nesneler, servis sanallaştırması, kuluçkalar, taklit uygulamalar ve sürücülerin kullanımını gerektirebilmekte ve **sistemin geri kalanından izole olarak yapılmaktadır**. Birim testleri fonksiyonaliteyi (ör. hesaplamaların doğruluğu), fonksiyonel olmayan özellikleri (ör. bellek sızıntılarını bulma) ve yapısal özellikleri (ör. karar testleri) kapsayabilir.

(components lar sistemin geri kalanından bağımsız olarak test edilirler.
Birimin fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması işlemidir.

Stubs(taklit uygulamalar) and drivers: önemli teknik bir ifade, sadece component teste ait bir özellik, bunu gördüğümüz zaman component test aklımıza gelecek.
coddan çıkarılan sonuçtan, sonuçtan çıkan kod)

Test esası

Birim testlerinde test esası olarak kullanılabilecek çalışma ürünlerinin örnekleri aşağıda verilmiştir:

- Detaylı tasarım
- Kod
- Veri modeli
- Birim gereksinimleri (spesifikasyonları) Test nesneleri

(Component testi yapabilmek için neler gerekli(test bases)

Component Test için elimizde mevcut bulunması gereken test baseler; detaylı bir design, kod , data modeli(developerların yazmış olduğu dataların saklandığı data modeller var, bu modlellerden bahsediyor), bu componenta ait özellikleri bilmemiz gerekiyor.)

Birim testlerinde tipik test nesneleri aşağıda verilmiştir:

- Birimler, bileşenler ve modüller
- Kod ve veri yapıları
- Sınıflar
- Veritabanı modülleri Tipik hatalar ve arızalar

(Component testte neleri test ediyoruz(test objects); Components, units, modules, Codes, data Structure, Classes, Database modules)

Birim testlerindeki yaygın hata ve arıza örnekleri aşağıda verilmiştir:

- Yanlış fonksiyonalite (örneğin, tasarım gereksinimlerinde açıklanandan farklı)
- Veri akışı problemleri
- Hatalı kod ve mantık

(Birim testinde yaygın hata ve arızlar(defect und failures); Incorrect functionality, Data flow problems, Incorrect code and logic)

Hatalar, genellikle tanımlı bir hata yönetim süreci olmadan, bulundukları anda giderilir. Bununla birlikte, yazılımcılar hataları raporlarsa, **kök neden analizi(root cause analyses, Grundursachenanalyse)** ve süreç iyileştirmesi(**process improvement, Prozessverbesserung**) için önemli bilgiler sağlanmış olur.

Birim testlerine özgü yaklaşımlar ve sorumluluklar

Birim testleri genellikle kodu yazan yazılımcı tarafından gerçekleştirilir, bu da test edilecek koda erişim gerektirir. Yazılımcılar bu seviyede geliştirme yapma ve hataların bulunup giderilmesini sağlama arasında gidip gelebilirler. Yazılımcılar genellikle bir birimin kodunu yazdıktan sonra testleri yazar ve yürütür. **Bununla birlikte, özellikle çevik yazılım geliştirmede, otomatikleştirilmiş birim test senaryoları yazmak uygulama kodunun yazılmasından önce gelebilir.**

Örneğin, test güdümlü yazılım geliştirme (TDD: test driven development, testgetriebene Entwicklung). Test güdümlü yazılım geliştirme oldukça döngüseldir ve otomatik test senaryoları yazma, sonrasında küçük kod parçaları oluşturma ve entegre etme, daha sonra birim testlerini yürütme, varsa sorunları giderme ve kodu yeniden düzenlemeyi içeren döngülere dayanır. Bu süreç, birim tamamen oluşturulana ve tüm birim testleri başarılı olana kadar devam eder. Test güdümlü geliştirme, önce-test-et yaklaşımının bir örneğidir. Her ne kadar test güdümlü geliştirme, ekstrem programlama (XP) ile ortaya çıkmış olsa da diğer çevik çerçevelerine (scrum, kanban) ve sıralı yaşam döngülerine yayılmıştır (bkz. ISTQB-AT Temel Seviye Çevik Test Uzmanı Devam Kursu Ders Programı).

***(Yazılımcılar genellikle bir birimin kodunu yazdıktan sonra testleri yazar ve yürütür.** Bununla birlikte, özellikle çevik yazılım geliştirmede, otomatikleştirilmiş birim test senaryoları(test case) yazmak uygulama kodunun(application code) yazılmasından önce gelebilir.)*

2.2.2 Entegrasyon Testi(Integration Testing, Integrationstest)

Entegrasyon testinin hedefleri

Entegrasyon testleri, birimler veya sistemler arasındaki etkileşimlere odaklanır. Entegrasyon testlerinin hedefleri aşağıda verilmiştir:

- Riskin azaltılması

- Arayüzlerin(interface, Schnittstellen) fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimde belirtildiği gibi olup olmadığının doğrulanması
- Arayüzlerin kalitesine dair güvenin oluşturulması
- Hataların bulunması (arayüzlerin kendisinde, birimlerde veya sistemlerde olabilir)
- Hataların daha üst test seviyelerine kaçmasının önlenmesi

(Intergrationtest; kod üzerinde yapılan değişikliklerin interface üzerinde herhangi bir arızaya neden olmadığından emin olmak için yapılır.component testte bir componenta odaklanılırken, integration testte birden fazla component veya sisteme odaklanıyoruz. Interface üzerinde yapılan test gördüğümüz zaman Integration test aklımıza gelmeli.

Enviroment değiştiği zaman, yeni component eklendiği zaman, beklenmeyen hatalara(unexepected defect) neden olup olmadğını kontrol etmek için integration test yapılır.)

Birim testlerinde olduğu gibi bazı durumlarda otomatikleştirilmiş **entegrasyon regresyon testler**i, değişikliklerin mevcut arayüzleri, birimleri veya sistemleri bozmadığına dair güvence sağlar.

Bu ders programında tanımlanan ve farklı büyüklükteki test nesneleri üzerinde gerçekleştirilebilen **iki farklı entegrasyon testi seviyesi vardır:**

- Birim entegrasyonu testleri(Component integration testing, Komponentenintegrationstests), entegre birimler arasındaki etkileşimlere ve arayüzlere odaklanır. Birim entegrasyonu testleri, birim testlerinden sonra gerçekleştirilir ve genellikle otomatikleştirilmiştir. Döngüsel ve artımlı yazılım geliştirmede(**iterative and incremental development, iterativen und inkrementellen Entwicklung)** birim entegrasyonu testleri genellikle sürekli entegrasyon sürecinin bir parçasıdır.
- Sistem entegrasyonu testleri(System integration testing, Systemintegrationstests), sistemler, paketler ve mikroservisler arasındaki etkileşimlere ve arayüzlere odaklanır.

(iki farklı Integration test seviyesi vardır; component integration test, system integration test. component integration test developerlar tarafından, system integration test testerlar tarafından yapılır.

sistem integration test sistem testten sonra veya sistem testle birlikte yapılır. önce yapılmaz.

sıralama bu şekilde : component test component integration test system test system integration test

Metinde between components, systems gibi çoğul ifade olursa, API, Interface gibi ifadeler geçerse bunun Integration test olduğunu anlarız. ACHTUNG!!)

Sistem entegrasyonu testleri, harici kurumlar (örneğin ağ servisleri) ile etkileşimleri ve onlar tarafından sağlanan arayüzleri de kapsayabilir. Bu durumda, yazılımı geliştirmekte olan kurumun harici arayüzler üzerinde bir kontrolü olmadığı için testlerde çeşitli zorluklar yaşanabilir (örneğin, harici kurumun kodundaki testi engelleyen hataların giderilmesi, test ortamı düzenlemeleri vb.). Sistem entegrasyonu

testleri, sistem testlerinden sonra veya devam eden sistem test faaliyetlerine paralel olarak yapılabilir (hem sıralı yazılım geliştirmede hem de döngüsel ve artımlı yazılım geliştirmede).

Test esası

Entegrasyon testlerinde test esası olarak kullanılabilecek çalışma ürünlerinin örnekleri aşağıda verilmiştir:

- Yazılım ve sistem tasarımı
- Sekans (dizi) diyagramı(Sequence diagrams, Sequenzdiagramme)
- Arayüz(**Interface, Schnittstellen**) ve iletişim protokolü gereksinimleri
- Kullanım senaryoları
- Birim veya sistem seviyesindeki mimari(Architecture at component or system level, Architektur auf Komponenten- oder Systemebene)
- İş akışları(Workflows, Workflows)
- Harici arayüz tanımları

(Integration testi yapabilmek için neler gerekli(test bases); interface, Sequence diagrams(Sekans dizi diyagramı), workflows, Architecture… bilinmesi gereken terimler.)

Test nesneleri

Entegrasyon testlerinde ele alınan tipik test nesneleri aşağıda verilmiştir:

- Alt-sistemler(Subsystems, Subsysteme)
- Veritabanları
- Altyapı(Infrastructure, Infrastruktur)
- Arayüzler
- API'ler
- Mikroservisler

(Integration testte neler test edilir(Test objects); Subsysteme, databases, Infrastrukture(altyapı), Interfaces, API, microservis)

Tipik hatalar ve arızalar

Birim entegrasyon testlerindeki tipik hata ve arıza örnekleri aşağıda verilmiştir:

- Hatalı veriler, eksik veriler veya hatalı veri kodlaması
- Arayüzlerin sıralamasında veya zamanlamasında hatalar
- Arayüz uyuşmazlığı
- Birimler arasındaki iletişim arızaları
- Birimler arasındaki giderilmemiş veya hatalı giderilmiş iletişim arızaları
- Birimler arasında iletimi yapılan verilerin anlamıyla, etkilediği birimlerle veya sınırları ile ilgili yanlış varsayımlar

Sistem entegrasyon testlerindeki tipik hata ve arıza örnekleri aşağıda verilmiştir:

- Sistemler arasında tutarsız mesaj yapıları
- Hatalı veriler, eksik veriler veya hatalı veri kodlaması
- Arayüz uyuşmazlığı
- Sistemler arasındaki iletişim arızaları
- Sistemler arasındaki giderilmemiş veya hatalı giderilmiş iletişim arızaları
- Sistemler arasında iletimi yapılan verilerin anlamıyla, etkilediği birimlerle veya sınırları ile ilgili yanlış varsayımlar
- Zorunlu güvenlik düzenlemelerine uyulmaması

Entegrasyon testlerine özgü yaklaşımlar ve sorumluluklar

Birim ve sistem entegrasyon testleri entegrasyonun kendisine odaklanmalıdır. Örneğin A modülü ile B modülü entegre ediliyorsa, testler modüllerin ayrı olarak fonksiyonalitesine değil, **modüller arasındaki iletişime odaklanmalıdır;** çünkü modüllerin fonksiyonalite testi, birim testleri sırasında ele alınmış olmalıdır. Aynı şekilde X sistemi ile Y sistemi entegre ediliyorsa, testler sistemlerin birbirinden bağımsız olarak fonksiyonalitesine değil, **sistemler arasındaki iletişime odaklanmalıdır**; çünkü sistemlerin fonksiyonalite testi, sistem testleri sırasında ele alınmış olmalıdır. **Entegrasyon test seviyesinde fonksiyonel ve fonksiyonel olmayan test çeşitleri uygulanabilir.**

Birim entegrasyon testleri genellikle yazılımcıların sorumluluğundadır. Sistem entegrasyon testleri ise genellikle test uzmanlarının sorumluluğundadır. İdeal olarak, sistem entegrasyon testlerini yapan **test uzmanları sistem mimarisini anlamalı ve entegrasyon planlamasında rol almış olmalıdır.**

Entegrasyon testleri ve entegrasyon stratejisi birimler veya sistemler oluşturulmadan önce planlanırsa, bu birimler veya sistemler, testin en verimli şekilde gerçekleştirilmesi için gereken sıraya göre oluşturulabilir.

Sistematik entegrasyon stratejileri, sistem mimarisine (örneğin yukarıdan aşağıya veya aşağıdan yukarıya(top-down and bottom-up)), **fonksiyonel görevlere, işlem dizilerine veya sistemin veya birimlerin başka bir sıralamasına dayanabilir.**

Hata bulma sürecini basitleştirmek ve hataları erkenden bulmak için entegrasyon "büyük patlama" (big bang)(tüm birimleri veya sistemleri tek bir seferde birleştirmek) **şeklinde değil, artımlı(incremental)** (bir seferinde az sayıda ek birim veya sistem) **bir şekilde olmalıdır**.

En karmaşık arayüzlerin risk analizi, **entegrasyon testlerini doğru alanlara odaklamaya yardımcı olabilir.**

Entegrasyonun **kapsamı ne kadar geniş olursa** belirli bir birim veya sistemdeki **hataları ayrıştırmak da o kadar zorlaşır ve bu da riskin artmasına ve sorun giderme için ek süreye neden olabilir.** Bu, yazılımın birim bazında birbiriyle entegre edildiği **sürekli entegrasyonun** (örneğin fonksiyonel entegrasyon (functional integration)) **yaygın bir uygulama haline gelmesinin nedenlerinden biridir.** Bu tür sürekli entegrasyon(continuous integration, kontinuierliche Integration) genellikle, ideal olarak birçok test seviyesinde gerçekleştirilen otomatikleştirilmiş regresyon testlerini içerir.

(Birim ve sistem entegrasyon testleri modüller arasındakive sistemler arasındaki integrationa odaklanmalıdır.

Birim entegrasyon testleri genellikle yazılımcıların sorumluluğundadır. Sistem entegrasyon testleri ise genellikle test uzmanlarının sorumluluğundadır.

Test uzmanları sistem mimarisini anlamalı ve entegrasyon planlamasında rol almış olmalıdır. Testlerin en verimli şekilde gerçekleştirilebilmesi için Entegrasyon testleri ve entegrasyon stratejisi birimler veya sistemler oluşturulmadan önce planlanmalı.

Sistematik entegrasyon stratejileri, sistem mimarisine(system architecture), fonksiyonel görevlere(functional tasks), işlem dizilerine(processing sequences), sistemin / birimlerin sıralamasına(aspect of the system or components) göre şekillenir.
Hata bulma sürecini basitleştirmek ve hataları erkenden bulmak için tüm birimleri veya sistemleri tek bir seferde birleştirmek "büyük patlama" (big bang) yerine artımlı(incremental) bir şekilde integration yapılmalıdır.

İlk olarak en karmaşık arayüzlerin risk analizine başlamak, entegrasyon testlerini doğru alanlara odaklamaya yardımcı olabilir.

Entegrasyonun kapsamının genişlemesi nisbetinde hatayı tespit zorlaşması, risklerin artması ve sorun gidermek için ek süreye ihtiyaç duyulmasına gibi nedenlerle sürekli entegrasyon(continuous integration, kontinuierliche Integration) (fonksiyonel entegrasyon)yaygın olarak kullanılmaktadır.

aşağıdakilerden hangisi integration test usullerindendir denildiğinde bu 4 ünü bilmemiz gerekli.
backbone, top-down, bottom-up, big bang)

2.2.3 Sistem Testi

Sistem testinin hedefleri

Sistem testleri, bütün bir sistemin veya ürünün davranış(behavior, Verhalten) ve yeteneklerine(capabilities, Fähigkeiten) **odaklanır** ve **genellikle sistemin gerçekleştirebileceği uçtan uca işleri (fonksiyonalite)(end-to-end tasks, End-to-End-Aufgaben) ve bu işleri gerçekleştirirken gösterdiği fonksiyonel olmayan davranışları ele alır.** Sistem testlerinin hedefleri aşağıda verilmiştir:

- Riskin azaltılması
- Sistemin fonksiyonel ve fonksiyonel olmayan davranışlarının tasarlandığı ve gereksinimlerde tanımlandığı gibi olup olmadığının doğrulanması
- Sistemin tamamlandığının ve beklendiği gibi çalışacağının sağlamasının yapılması
- Bir bütün olarak sistemin kalitesine dair güven oluşturulması
- Hataların bulunması
- Hataların daha üst test seviyelerine veya canlıya kaçmasının önlenmesi

Bazı sistemler için veri kalitesini doğrulamak bir hedef olabilir. Birim testlerinde ve entegrasyon testlerinde de olduğu gibi, bazı durumlarda otomatikleştirilmiş sistem regresyon testleri yapılan değişikliklerin mevcut yazılım özelliklerini veya yazılımın uçtan uca yetkinliklerini bozmadığına dair güvence sağlar. **Sistem testleri genellikle paydaşlar tarafından sürüm kararları almak için kullanılan bilgileri üretir**. Sistem testleri ayrıca yasal veya sözleşmeye dayalı gereksinimleri veya standartları da karşılayabilir.

(Sistem testleri, bütün bir sistemin veya ürünün davranış(behavior, Verhalten) ve yeteneklerine(capabilities, Fähigkeiten) odaklanır ve genellikle sistemin gerçekleştirebileceği uçtan uca

işleri (fonksiyonalite)(end-to-end tasks, End-to-End-Aufgaben) ve bu işleri gerçekleştirirken gösterdiği fonksiyonel olmayan davranışları(kullanılabilirlik, performans, güvenlik) ele alır.

system test ile üretilen bilgiler stakeholderlar tarafından o ürünün product aşamasına geçip geçmeyeceğine karar vermeleri için kullanılır. System veya fonksiyonel end-to-end yazarsa system test, systems yazarsa integration test olduğunu anlamalıyız.)

Sistem test ortamı ideal olarak canlı ortama karşılık gelmelidir.

Test esası

Sistem testlerinde test esası olarak kullanılabilecek çalışma ürünlerinin örnekleri aşağıda listelenmiştir:

- Sistem ve yazılım gereksinimleri (fonksiyonel ve fonksiyonel olmayan)
- Risk analizi raporları
- Kullanım senaryoları
- Epikler ve kullanıcı hikâyeleri
- Sistem davranış modelleri
- Durum diyagramları
- Sistem ve kullanıcı kılavuzları

(System testi yapabilmek için elimizde olması gerekenler; requirement, Risk analysis reports, Test cases, Epic ve Usert stroy, sistem davranış modelleri, durum diyagramları(State diagrams, Zustandsdiagrame)), sistem ve kullanıcı klavuzları)

Test nesneleri

Sistem testlerinde ele alınan tipik test nesneleri aşağıda listelenmiştir:

- Uygulamalar
- Donanım/yazılım sistemleri
- İşletim sistemleri
- Test edilen sistem (SUT
- Sistem yapılandırması ve yapılandırma verileri

(System testinde neleri test ediyoruz; Applications, hardware / Software system, işletim sistemleri Operating system(Betriebsystem), Test edilen sistem(system under Test(SUT), systeme unter Test), system configuration, configuration data)

Tipik hatalar ve arızalar

Sistem testlerindeki tipik hata ve arıza örnekleri aşağıda listelenmiştir:

- Hatalı hesaplamalar
- Hatalı veya beklenmeyen sistem fonksiyonel veya fonksiyonel olmayan davranışı
- Sistem içindeki hatalı kontrol ve/veya veri akışları
- Fonksiyonel uçtan uca işlerin olması gerektiği gibi ve eksiksiz bir şekilde gerçekleştirilememesi
- Canlıda sistemin doğru şekilde çalışmaması
- Sistemin, sistem ve kullanıcı kılavuzlarının tanımlandığı şekilde çalışmaması

Spesifik yaklaşımlar ve sorumluluklar

Sistem testleri, sistemin bir bütün olarak uçtan uca (fonksiyonel ve fonksiyonel olmayan) **davranışına odaklanmalıdır. Sistem testleri**, test edilecek **sisteme özgü en uygun teknikleri kullanmalıdır** (bkz. Bölüm 4). Örneğin, fonksiyonel özelliklerin gereksinimlerde tanımlandığı şekilde olup olmadığını doğrulamak için bir karar tablosu(decision table, Entscheidungstabelle) oluşturulabilir.

Sistem testlerini genellikle bağımsız test uzmanları gerçekleştirir. Gereksinimlerdeki hatalar (örneğin; eksik kullanıcı hikâyeleri, yanlış belirlenen kullanıcı gereksinimleri vb.) beklenen sistem davranışının anlaşılamamasına veya yanlış anlaşılmasına yol açabilir. Bu tür durumlar, yanlış pozitif ve yanlış negatiflere neden olabilir; bu da, zaman kaybına ve hata bulma etkinliğinin azalmasına neden olur. Test uzmanlarının kullanıcı hikâyesinin geliştirilmesi veya gözden geçirilmesi gibi statik test aktivitelerine erken aşamalarda katılmaları bu gibi durumların görülme sıklığını azaltır.

(Test uzmanlarının user story'lerin geliştirilmesi veya gözden geçirilmesi gibi statik test aktivitelerine erken aşamalarda katılmaları requirementlar'daki veya user story'lerdeki hatalardan kaynaklı yanlış pozitif ve yanlış negatiflerin önüne geçebilir, buda hataların erken farkedilerek erken aşamalarda düzeltilmesini sağlar. Sistem testlerini genellikle bağımsız test uzmanları gerçekleştirir)

2.2.1 Kabul Testi(Acceptance Testing, Abnahmetest)

Kabul testinin hedefleri

Sistem testleri gibi kabul testleri de genellikle bütün bir sistemin veya ürünün davranışına ve yeteneklerine odaklanır. **Kabul testlerinin hedefleri** aşağıda verilmiştir:

- Bir bütün olarak sistemin kalitesine dair güven oluşturulması
- Sistemin tamamlandığının ve beklendiği gibi çalışacağının sağlamasının yapılması
- Sistemin fonksiyonel ve fonksiyonel olmayan davranışlarının gereksinimlerde belirtildiği gibi olup olmadığının doğrulanması

Kabul testleri, sistemin müşteriye (son kullanıcı) çıkmaya ve kullanıma hazır olduğunu değerlendirmeye yönelik bilgiler elde etmek için yapılır. Kabul testleri sırasında hatalar bulunabilir, ancak **hata bulmak genellikle kabul testinin bir amacı değildir** ve kabul testleri sırasında önemli sayıda hata bulunması bazı durumlarda büyük bir proje riski olarak kabul edilebilir. Kabul testleri ayrıca yasal veya sözleşmeye dayalı gereksinimleri veya standartları da karşılayabilir.

(Kabul Testi(Acceptance Testing, Abnahmetest); hata bulmak için yapılmaz, sistemin end user'a çıkmasına ve kullanıma hazır olup olmadığına bakılır.Beklendiği gibi çalışacağının değerlendirmesi için yapılır. Sistemin kalitesine dair güven oluşturmaya yarar. Fonksiyonel ve fonksiyonel olmayan davranışların requirementlarda tanımlanan gibi olup olmadığının doğrulamasını yapmaya yarar.)

Yaygın kullanılan kabul testi çeşitleri arasında aşağıdakiler yer alır:

(4 çeşit Acceptance Test vardır)

- Kullanıcı kabul testleri(User acceptance testing, Benutzerabnahmetest)
- Operasyonel kabul testleri(Operational acceptance testing, Betrieblicher Abnahmetest)
- Sözleşmeye dayalı ve yasal kabul testleri(Contractual and regulatory acceptance testing, Vertraglicher und regulatorischer Abnahmetest)
- Alfa ve beta testleri(Alpha and beta testing, Alpha- und Beta-Test)

Bunları her biri aşağıdaki dört alt bölümde açıklanmaktadır.

Kullanıcı kabul testleri(User acceptance testing, Benutzerabnahmetest)

Sistemin kabul testlerinin **kullanıcılar tarafından yapılmasıdır**. Genellikle gerçek veya simüle edilmiş bir operasyonel ortamda sistemin hedeflenen kullanıcıların kullanımına uygunluğunun belirlenmesi için yapılır. **Temel amaç**, **kullanıcıların** ihtiyaçlarını karşılamak, gereksinimleri yerine getirmek ve istenilen iş süreçlerini gerçekleştirmek için sistemi **asgari zorluk, maliyet ve riskle kullanabileceklerine dair güven oluşturmaktır.**

(User acceptance Test kullanılcılar tarafından yapılır. Temel amaç; Kullanıcıların ihtiyaçlarını karşıladığına, sistemi asgari zorluk, maliyet ve riskle kullanabileceklerine dair güven oluşturmaktır)

Operasyonel kabul testleri(Operational acceptance testing, Betrieblicher Abnahmetest)

Operasyonel kabul testleri operatörler veya sistem yöneticileri tarafından, genellikle simüle edilmiş canlı ortamda gerçekleştirilir. Testler operasyonel konulara odaklanır ve aşağıdakileri içerebilir:

- Yedekleme ve geri yükleme testleri
- Kurma, kaldırma ve yükseltme
- Felaket kurtarma
- Kullanıcı yönetimi
- Bakım işleri
- Veri yükleme ve taşıma işleri
- Güvenlik açıkları için kontroller
- Performans testleri

Operasyonel kabul testlerinin temel amacı, operatörlerin veya sistem yöneticilerinin sistemi istisnai veya zorlu koşullar altında bile düzgün bir şekilde çalışmaya devam ettirebileceği konusunda güven oluşturmaktır.

(İşletmedeki çalışanlar veya sistem yöneticileri tarafından yapılır. Sistemin operasyonel özelliklerine odaklanılır. Yedekleme, install, uninstall, upgrade, user management, maintenance test, performans test gibi testler yapılır. Testin amacı; sistemin istisnai koşullar altında bile düzgün bir şekilde çalıştığı konusunda güven oluşturmaktar.)

Sözleşmeye dayalı ve yasal kabul testleri(Contractual and regulatory acceptance testing, Vertraglicher und regulatorischer Abnahmetest)

Sözleşmeye dayalı kabul testleri, özel olarak geliştirilmiş bir yazılım üretmek için sözleşmenin kabul kriterlerine göre gerçekleştirilir. Kabul kriterleri, taraflar sözleşmeyi kabul ederken tanımlanmış olmalıdır. Sözleşme kabul testleri genellikle **kullanıcılar veya bağımsız test uzmanları tarafından gerçekleştirilir.**

Yasal kabul testleri, hükümet, yasal veya emniyet düzenlemeleri gibi uyulması gereken düzenlemelere göre yapılır. Yasal kabul testleri genellikle kullanıcılar veya bağımsız test uzmanları tarafından gerçekleştirilir, bazen yasal kurumların sonuçlara şahitlik etmesi veya sonuçları denetlemesi gerekebilir.
Sözleşmeye dayalı ve yasal kabul testlerinin temel amacı, sözleşmeye veya yasal düzenlemelere uyumluluk sağlandığına dair güven oluşturmaktır.

(kullanıcılar veya bağımsız test uzmanları tarafından gerçekleştirilir. Testin amacı; Sistemin sözleşmeye(kontrata), yasalara veya emniyet düzenlemelerine uyumluluk sağlandığına dair güven oluşturmaktır.)

Alfa ve beta testleri(Alpha and beta testing, Alpha- und Beta-Test)

Alfa ve beta testleri genellikle **yazılım ürünü piyasaya sürülmeden önce potansiyel veya mevcut kullanıcılar, müşteriler ve/veya operatörlerden geri bildirim almak isteyen ticari paket yazılımın (COTS) geliştiricileri tarafından kullanılır. Alfa testleri yazılım geliştiren kuruluşun tesislerinde**, sadece **yazılım geliştirme ekibi** tarafından değil, **potansiyel veya mevcut müşteriler** ve/veya **operatörler** veya **bağımsız bir test ekibi tarafından** gerçekleştirilir. **Beta testleri** ise **potansiyel veya mevcut müşteriler** ve/veya **operatörler** tarafından **kendi ortamlarında yapılır. Beta testleri alfa testlerinden sonra gelebilir veya öncesinde hiç alfa testi yapılmadan da gerçekleştirilebilir.**

Alfa ve beta testlerinin hedeflerinden biri, kullanıcılarda sistemi normal, günlük koşullar altında ve operasyonel ortam(lar)da hedeflerine asgari zorluk, maliyet ve riskle ulaşmak için kullanabileceklerine dair güven oluşturmaktır. Diğer bir hedef ise sistemin kullanılacağı şartlar ve ortam(lar) ile ilgili hataların, özellikle bu koşullar ve ortam(lar)ın yazılım geliştirme ekibi tarafından aynısının oluşturulmasının zor olduğu durumlar için belirlenmesi olabilir.

(Alfa ve Beta Testleri yazılım piyasaya sürülmeden önce yapılır.
Alfa Testi; Yazılımı geliştiren kuruluşun tesislerinde gerçekleştirilir. Kullanıcılar, developerlar, opertaörler ve test ekibi tarafından gerçekleştirilir.

Beta testi; Kullanıcılar veya operatörler tarafından kendi ortamlarında gerçekleştirilir.

Önce alfa test yapılır sonra Beta test gerçekleştirilir. Alfa test yapılmaması durumunda sadece Beta testte yapılabilir

Alfa ve Beta testlerinin hedefi kullanıcılarda sisteme karşı güven oluşturmak, sistemin kullanılacağı şartlar ve ortamlarda çıkabilecek muhtemel hataları tesbit etmektir)

Test esası

Herhangi bir kabul testi için test esası olarak kullanılabilecek çalışma ürünlerinin örnekleri aşağıda listelenmiştir:

- İş süreçleri
- Kullanıcı veya iş gereksinimleri
- Düzenlemeler, yasal sözleşmeler ve standartlar
- Kullanım senaryoları
- Sistem gereksinimleri
- Sistem veya kullanıcı dokümantasyonu
- Kurulum prosedürleri
- Risk analizi raporları

Ek olarak, operasyonel kabul testleri için test senaryolarının türetilmesinde test esası olarak aşağıdaki çalışma ürünlerinden bir veya daha fazlası kullanılabilir:

- Yedekleme ve geri yükleme prosedürleri
- Felaket kurtarma prosedürleri
- Fonksiyonel olmayan gereksinimler
- Operasyon dokümantasyonu
- Dağıtım ve kurulum talimatları
- Performans hedefleri
- Veritabanı paketleri
- Güvenlik standartları veya düzenlemeleri

Tipik test nesneleri

Herhangi bir kabul testinde(**Acceptence test**) ele alınabilecek tipik test nesneleri aşağıda verilmiştir:

- Test edilen sistem (SUT)
- Sistem yapılandırması ve yapılandırma verileri
- Tamamen entegre edilmiş bir sistem için iş süreçleri
- Kurtarma sistemleri ve hayati önem taşıyan alanlar (iş sürekliliği ve felaket kurtarma testleri için)
- Operasyonel süreçler ve bakım süreçleri
- Formlar
- Raporlar
- Üretim verileri

Tipik hatalar ve arızalar

Herhangi bir kabul testi için tipik hata örnekleri aşağıda verilmiştir:

- Sistem iş akışlarının iş veya kullanıcı gereksinimlerini karşılamaması
- İş kurallarının doğru şekilde uygulanmaması
- Sistemin sözleşmeden kaynaklanan veya yasal gereksinimleri karşılamaması
- Güvenlik açıkları, fazla yük altında yetersiz performans veya desteklenen bir platformda yanlış çalışma gibi fonksiyonel olmayan arızalar

Kabul testine özgü yaklaşımlar ve sorumluluklar

Kabul testleri genellikle müşterilerin, kullanıcıların, ürün sahiplerinin veya sistem operatörlerinin sorumluluğundadır ve diğer paydaşlar da bu sürece katılabilir.

Kabul testleri genellikle **sıralı bir yazılım geliştirme yaşam döngüsündeki en son test seviyesi** olarak düşünülür, ancak aşağıdaki örneklerde olduğu gibi, başka zamanlarda da gerçekleştirilebilir:

- Bir ticari paket yazılımın (COTS) kabul testleri, kurulum yapıldığında veya diğer sistemlere entegre edildiğinde gerçekleştirilebilir.
- Yeni bir fonksiyonel geliştirmenin kabul testleri sistem testlerinden önce yapılabilir.

Döngüsel yazılım geliştirmede, proje ekipleri her döngü sırasında ve döngünün sonunda çeşitli kabul testleri uygulayabilir: **yeni bir özelliği kabul kriterlerine karşı doğrulamaya odaklananlar** ve yeni bir özelliğin kullanıcıların ihtiyaçlarını karşıladığının sağlamasını yapmaya odaklananlar gibi. Ek olarak, **alfa testleri ve beta testleri, her döngünün sonunda, her döngünün tamamlanmasından sonra veya bir dizi döngüden sonra yapılabilir**. Kullanıcı kabul testleri, operasyonel kabul testleri, yasal kabul testleri ve sözleşme kabul testleri de her döngünün kapanışında, her döngünün tamamlanmasından sonra veya bir dizi döngüden sonra yapılabilir.

(Kabul testleri(Acceptence Test) genellikle müşterilerin, kullanıcıların, ürün sahiplerinin veya sistem operatörlerinin sorumluluğundadır ve diğer paydaşlar da bu sürece katılabilir.

Sıralı yazılım geliştirme (sequential development, sequenziellen Entwicklungslebenszyklus) yaşam döngüsündeki en son test seviyesidir istisna olarak ticari paket yazılımın(COTS) kabul testleri kurulum yapıldığında ve diğer sistemlere entegre edildiğinde gerçekşelştirilir.

Döngüsel yazılım geliştirmede(iterative development, iterativen Entwicklung), proje ekipleri her döngü sırasında ve döngünün sonunda, yeni bir özelliği kabul kriterlerine karşı doğrulamada ve requirementleri karşılayıp karşılamadağının doğrulanması durumlarında yapılır.

alfa testleri ve beta testleri, her döngünün sonunda, her döngünün tamamlanmasından sonra yapılabilir)

2.3 Test Çeşitleri(Test Types, Testarten)

Test çeşidi(Test Types, Testart), **belirlenmiş test hedeflerine dayanarak bir yazılımın belirli özelliklerini veya bir sistemin bir bölümünü test etmeyi amaçlayan bir test aktiviteleri grubudur**. Bu test hedefleri(Test objectives, Testziele) aşağıdakileri içerebilir:

- Bütünlük, doğruluk ve uygunluk gibi **fonksiyonel kalite özelliklerinin** değerlendirilmesi
- Güvenilirlik, performans, güvenlik, uyumluluk ve kullanılabilirlik gibi **fonksiyonel olmayan kalite özelliklerinin** değerlendirilmesi

- Birim veya sistemin **yapısının veya mimarisinin** doğru, eksiksiz ve gereksinimlerde belirtildiği gibi olup olmadığının değerlendirilmesi
- **Değişikliklerin etkilerinin değerlendirilmesi**: hataların giderildiğini onaylama (onaylama testleri) ve düzeltilen hatanın veya **yapılan değişikliğin istenmeyen değişiklikleri tetikleyip tetiklemediğini bulma (regresyon testleri) gibi.**

(Test hedefleri; fonksiyonel ve fonksiyonel olmayan kalite özelliklerinin değerlendirilmesi, yapısının ve mimarisinin doğru, eksiksiz ve gereksinimlerde tanımlandığı olup olmadığının değerlendirilmesi, hataların giderildiğini onaylama ve düzeltilen hatanın veya yapılan değişikliğin istenmeyen değişiklikleri tetikleyip tetiklemediğini bulma(regresyon testleri))

2.3.1 Fonksiyonel Testler

Bir sistemin fonksiyonel testleri(functional testing, Funktionale Tests), **sistemin gerçekleştirmesi gereken fonksiyonları değerlendiren testleri içerir**. Fonksiyonel gereksinimler; iş gereksinimleri, kullanıcı gereksinimleri, epikler, kullanıcı hikâyeleri, kullanım senaryoları veya fonksiyonel spesifikasyonlar gibi çalışma ürünlerinde tanımlanmış olabilir. **Fonksiyonlar gereksinimler sistemin ne yapması gerektiğini tanımlar.**

Fonksiyonel testler tüm test seviyelerinde yapılmalıdır (örneğin; birim seviyesindeki fonksiyonel testler ilgili birimin gereksinimlerine dayandırılmalıdır), ancak odağı her seviyede farklıdır (bkz. bölüm 2.2).

Fonksiyonel testler yazılımın davranışını göz önüne alır; bu nedenle, birim veya sistemin fonksiyonalitesi için **test koşullarını ve test senaryolarını oluşturmada kara kutu teknikleri kullanılabilir** (bkz. bölüm 4.2).

Fonksiyonel testlerin bütünlük derecesi, fonksiyonel kapsam ile ölçülebilir. **Fonksiyonel kapsam**, fonksiyonel gereksinimin testlerle ne kadar ele alındığını belirtir ve **gereksinimin yüzdesi** olarak ifade edilir. Testler ve fonksiyonel gereksinimler arasındaki izlenebilirlik kullanılarak bu **gereksinimlerin testlerde ele alınan yüzdesi hesaplanabilir ve böylece potansiyel olarak kapsam eksiklikleri belirlenebilir**.

Fonksiyonel test tasarımı ve koşumu, yazılımın çözdüğü belirli bir iş probleminin bilgileri (örneğin, petrol ve gaz endüstrileri için jeolojik modelleme yazılımı) veya yazılımın hizmet ettiği özel roller (örneğin, etkileşimli eğlence sağlayan bilgisayar oyun yazılımı) gibi özel bilgi veya becerileri içerebilir.

(Fonksiyonel testler yazılımın kendisini değil davranışını inceler, kodları görmemiz gerekmez, bu nedenle Black box teknikleri test için kullanılabilir. Sistem çalıştıktan sonra beklenen neticeleri üretip üretmediğini test eden test tipine Functional Test denir. Sistem ne yapmalı sorusunun cevabını functional test verir(nasıl yaptığını değil). Fonksiyonel testler tüm test seviyelerinde yapılmalıdır.

Fonksiyonel kapsam, gereksinimin yüzdesi olarak ifade edilir. Gereksinimler ve bu gereksinimlerin testlerde ele alınan yüzdesi hesaplanabilir. Böylece potansiyel olarak kapsam eksiklikleri belirlenebilir)

2.3.2 Fonksiyonel Olmayan Testler(Non-functional Testing, Nicht-funktionale Tests)

Bir sistemin fonksiyonel olmayan testleri(Non-functional Testing, Nicht-funktionale Tests), **sistemlerin ve yazılımların, kullanılabilirlik, performans veya güvenlik gibi özelliklerini değerlendirir.** Yazılım ürün kalitesi özelliklerinin sınıflandırması için ISO standardına (ISO/IEC 25010) bakınız. **Fonksiyonel olmayan testler sistemin yapılması gerekenleri "ne kadar iyi" yaptığını ölçümlemeye çalışır.**

Yaygın yanlış bilgilerin aksine fonksiyonel olmayan **testler tüm test seviyelerinde ve sıkça gerçekleştirilmeli ve mümkün olduğunca erken yapılmalıdır.** Fonksiyonel olmayan **hataların geç fark edilmesi bir projenin başarısı için son derece tehlikeli olabilir.**

Fonksiyonel olmayan testler için **test koşullarını ve test senaryolarını üretmekte kara kutu test teknikleri** (bkz. Bölüm 4.2) **kullanılabilir**. Örneğin, **performans testleri için stres koşullarını belirlemede sınır değer analizi(boundary value analysis, Grenzwertanalyse) kullanılabilir.**

Fonksiyonel olmayan testlerin **bütünlük derecesi, fonksiyonel olmayan kapsam ile ölçülebilir**. Fonksiyonel olmayan kapsam, **fonksiyonel olmayan bir gereksinimin testlerle ne ölçüde ele alınmış olduğunu belirtir ve kapsanan gereksinimin bir yüzdesi olarak ifade edilir**. Örneğin, bir mobil uygulama için testler ve desteklenen cihazlar arasındaki izlenebilirlik kullanılarak uyumluluk testleriyle ele alınan cihazların yüzdesi hesaplanabilir ve **böylece potansiyel olarak kapsam eksiklikleri belirlenebilir.**

Fonksiyonel olmayan test tasarımı(design) ve koşumunda(execute), bir tasarım veya teknolojinin yapısal açıdan zayıf yönleri (örneğin, belirli programlama dilleriyle ilgili güvenlik açıkları) veya belirli bir kullanıcı tabanı (örneğin hastane yönetim sistemi kullanıcı grupları) hakkında bilgiler gibi özel beceriler veya bilgiler yer alabilir.

(fonksiyonel olmayan testler(Non-functional Testing, Nicht-funktionale Tests), sistemlerin ve yazılımların, kullanılabilirlik, performans veya güvenlik gibi özelliklerini "ne kadar iyi" yaptığını ölçümlemeye çalışır. Testler tüm test seviyelerinde ve sıkça gerçekleştirilmeli ve mümkün olduğunca erken yapılmalıdır. Fonksiyonel olmayan hataların geç fark edilmesi bir projenin başarısı için son derece tehlikeli olabilir. Test koşullarını ve test senaryolarını üretmekte kara kutu test teknikleri kullanılabilir. Performans testleri için stres koşullarını belirlemede sınır değer analizi(boundary value analysis, Grenzwertanalyse) kullanılabilir.

Fonksiyonel olmayan kapsam, gereksinimin yüzdesi olarak ifade edilir. Gereksinimler ve bu gereksinimlerin testlerde ele alınan yüzdesi hesaplanabilir. Böylece potansiyel olarak kapsam eksiklikleri belirlenebilir)

2.3.3 Beyaz Kutu Testleri(White-box Testing, White-Box-Tests)

Beyaz kutu testleri(White-box Testing, White-Box-Tests), **sistemin iç yapısına dayanan testleri oluşturur.** İç yapı(Internal structure, Interne Structure); kod, mimari(architecture),

iş akışları(work flows) ve/veya sistem içindeki veri akışlarını(data flows, Datenflüsse) **içerebilir** (bkz. Bölüm 4.3).

Beyaz kutu testlerinin bütünlük derecesi yapısal kapsam(structural coverage, strukturelle Überdeckung) **ile ölçülebilir. Yapısal kapsam**, bir yapısal öğenin testlerle ne ölçüde ele alınmış olduğunu belirtir ve **kapsanan öğenin yüzdesi olarak ifade edilir.**

Birim testi seviyesinde, kod kapsamı(code coverage, Codeüberdeckung) test edilen birim kodunun yüzdesine dayanır ve **birimde test edilen komutların yüzdesi veya test edilen kararların yüzdesi gibi kodun farklı yönleri (kapsam öğeleri) açısından ölçülebilir.** Bu kapsam çeşitlerine toplu olarak **kod kapsamı(code coverage, Codeüberdeckung**) denir**.** Birim entegrasyon testleri seviyesinde beyaz kutu testleri, birimler arasındaki arayüzler gibi sistem mimarisine dayanabilir ve **yapısal kapsam(structural coverage, strukturelle Überdeckung), testler ile ele alınmış arayüzlerin yüzdesi olarak ölçülebilir.**

Beyaz kutu test tasarımı ve koşumu, kodun yazılma şekli (örneğin kod kapsamı araçlarını kullanmak), verilerin nasıl depolandığı (örneğin olası veritabanı sorgularını değerlendirmek), kapsam araçlarının nasıl kullanılacağı ve sonuçlarının doğru şekilde yorumlanması gibi özel bilgi veya becerileri içerebilir.

(Beyaz kutu testleri(White-box Testing, White-Box-Tests), sistemin iç yapısına dayanan testleri oluşturur. Structure, architecture, implementation, design, code, work flow(if), data, data flows gibi terimleri gördüğümüz zaman white box Test aklımıza gelmeli. Kodlar görülmeden white box test yapılamaz.

Beyaz kutu testlerinin bütünlük derecesi yapısal kapsam(structural coverage, strukturelle Überdeckung) ile ölçülebilir. Yapısal kapsam, kapsanan öğenin yüzdesi olarak ifade edilir.

Test edilen birim kodunun yüzdesine kod kapsamı(code coverage, Codeüberdeckung) denir)

2.3.4 Değişiklikle İlgili Testler(Change-related Testing, Änderungsbezogenes Testen)

Bir sistemde, bir **hatayı çözmek** için veya **yeni veya değişen fonksiyonalite** nedeniyle **değişiklikler yapıldığında**, değişikliklerin **hatayı çözdüğünü** veya **fonksiyonaliteyi doğru bir şekilde** hayata geçirdiğini ve öngörülemeyen **olumsuz sonuçlara neden olmadığını doğrulamak için testler yapılmalıdır.**

Onaylama testleri(Confirmation testing, Fehlernachtests): Bir hata çözüldükten sonra, hata nedeniyle başarısız olmuş tüm test senaryoları tekrar test edilebilir; bu testler yazılımın yeni versiyonunda yeniden koşturulmalıdır. Hatanın fonksiyonalite eksikliğinden kaynaklanması durumunda, yazılımın test edilmesi için yeni testler de yazılabilir. En azından hatadan kaynaklanan arızayı/arızaları veya eksikliği yeniden oluşturmak için gereken test adımları, yazılımın yeni versiyonunda tekrar koşturulmalıdır. **Onaylama testinin amacı(Confirmation Testing, Fehlernachtests), asıl hatanın başarıyla çözülüp çözülmediğini onaylamaktır.**

Regresyon testleri(Regression testing, Regressionstests)**: Kodun bir bölümünde yapılan bir değişikliğin** (bir düzeltme veya başka bir değişiklik çeşidi olabilir) **kazara kodun diğer bölümlerinin** (aynı birim içinde, aynı sistemin diğer birimlerinde ve hatta diğer sistemlerde) **davranışını olumsuz bir şekilde etkilemesi** olasıdır. Değişiklikler, işletim sisteminin veya veritabanı yönetim sisteminin yeni bir versiyonu gibi ortamda yapılan değişiklikler de olabilir. **Bu istenmeyen yan etkilere regresyon denir.** Regresyon testleri, **bu gibi istenmeyen yan etkileri bulmak için yapılan testleri içerir.**

Onaylama testleri ve regresyon testleri tüm test seviyelerinde yapılabilir.

Özellikle döngüsel(**iterative**) ve artımlı(**incremental**) yazılım geliştirme yaşam döngülerinde (ör. Çevik), yeni özellikler, **mevcut özelliklerde yapılan değişiklikler** ve kod yeniden düzenlemesi (refactoring, Code-Restrukturierung), **kodda sık sık değişiklik yapılmasıyla sonuçlanır**; **bu da değişiklikle ilgili testler gerektirir.** Sistemin evrilen yapısı nedeniyle Bu durum nesnelerin (örneğin aygıtların) sıklıkla güncellendiği veya başkasıyla değiştirildiği Nesnelerin İnterneti sistemleri için özellikle önemlidir.

Regresyon testi grupları yazılım geliştirme yaşam döngüsü içinde birçok kez koşturulur ve genellikle fazla değişikliğe uğramazlar, bu nedenle regresyon testleri otomasyon için güçlü bir adaydır. Bu testlerin otomasyonu projenin erken aşamalarında başlamalıdır (bkz. Bölüm 6).

(Change related denilince confirmation ve regression test aklımıza gelmeli. iki adet Değişiklikle İlgili Test(Change-related Testing, Änderungsbezogenes Testen) vardır.

Onaylama testi(Confirmation Testing, Fehlernachtests), asıl hatanın başarıyla çözülüp çözülmediğini onaylamak için yapılır.

Regresyon testleri(Regression testing, Regressionstests): Kodun bir bölümünde yapılan bir değişiklik kazara kodun diğer bölümlerinin davranışını olumsuz şekilde etkileyebilir.Bu gibi istenmeyen yan etkileri bulmak için yapılan testlere regresyon testi denir.

Özellikle Agile gibi software life cycle'larda kodlarda sürekli bir değişiklik, kodların yeniden düzenlenmesi(refactoring, Code-restructurierung) söz konusu olduğundan confirmation ve regresyon testleri bu yaşam döngüsünde çok önemlidir)

2.3.5 Test Çeşitleri ve Test Seviyeleri(Test Types and Test Levels, Testarten und Teststufen)

Yukarıda belirtilen **test çeşitlerinden herhangi birini tüm test seviyelerinde gerçekleştirmek mümkündür.** Örnek olarak, fonksiyonel testlerle başlayarak bir bankacılık uygulaması için tüm test seviyelerinde fonksiyonel, fonksiyonel olmayan, beyaz kutu ve değişiklikle ilgili testlerin örnekleri verilecektir:

Fonksiyonel testler için örnekler:

- Birim testleri(Component testing) için, bir birimin bileşik faizi **doğru hesaplayıp hesaplamadığı** test edilebilir.
- Birim entegrasyon testleri(Integration testing) için, kullanıcı arayüzünde kaydedilen hesap bilgilerinin **iş mantığına nasıl aktarıldığı** test edilebilir.
- Sistem testleri için, müşterilerin vadesiz hesaplarını kullanarak bir krediye nasıl **başvurabilecekleri** test edilebilir.
- Sistem entegrasyon testleri için, müşterinin kredi puanını kontrol etmek için sistemin **harici bir mikroservisini** nasıl **kullandığı** test edilebilir.
- Kabul testleri için, bankacının bir **kredi başvurusunu** nasıl **onayladığı** veya reddettiği test edilebilir.

Fonksiyonel olmayan testler için örnekler:

- Birim testleri için, karmaşık bir faiz hesaplaması yapmak için gereken işlemci (CPU) **işlem sayısı** test edilebilir.
- Birim entegrasyon testleri için, kullanıcı arayüzünden iş mantığına iletilen verilerden kaynaklanan arabellek aşımı **güvenlik açıkları** test edilebilir.
- Sistem testleri için, sunum katmanının tüm desteklenen tarayıcılarda ve mobil cihazlarda çalışıp çalışmadığı taşınabilirlik testleri(portability tests, Übertragbarkeitstests) yapılarak test edilebilir.
- Sistem entegrasyon testleri için, kredi puanı mikroservisinin yanıt vermemesi durumunda, sistem sağlamlığını değerlendirmek üzere **güvenilirlik testleri** yapılabilir.
- Kabul testleri için, bankacının kredi işleme arayüzünün **engelliler tarafından erişilebilirliğini değerlendirmek** üzere kullanılabilirlik testleri yapılabilir.

(Taşınabilirlik testleri(portability tests, Übertragbarkeitstests): Sistem testleri için sunum katmanının tüm desteklenen tarayıcılarda ve mobil cihazlarda çalışıp çalışmadığını test eder, güvenlik açıkları var mı, yok mu buna bakar vs.)

Beyaz kutu testler(White-box testing) için örnekler:

- Birim testleri için, finansal hesaplamalar yapan tüm birimlerde **tam komut** ve karar kapsamı (bkz. Bölüm 4.3) elde etmek için testler yapılabilir.
- Birim entegrasyon testleri için, tarayıcı arayüzündeki her ekranın bir sonraki ekrana ve iş mantığına **verileri nasıl ilettiğini denemek** için testler yapılabilir.
- Sistem testleri için, bir kredi başvurusu sırasında müşterinin karşısına çıkabilecek **web sayfalarının sıralamasını** kapsayacak şekilde testler yapılabilir.
- Sistem entegrasyon testleri için, kredi puanı mikroservisine gönderilen **tüm olası sorgulama çeşitlerini** denemek üzere testler yapılabilir.
- Kabul testleri için, tüm desteklenen finansal **veri dosyası yapılarını** ve bankadan bankaya transferler için değer aralıklarını kapsayacak şekilde testler yapılabilir.

Değişiklikle İlgili Testler(Change-related Testing, Änderungsbezogenes Testen) için örnekler:

- Birim testleri için, her birimin otomatikleştirilmiş **regresyon testleri** oluşturulur ve sürekli entegrasyon yapısına dâhil edilir.
- Birim entegrasyonu testleri için, kodda yapılan düzeltmeler kod deposuna eklenirken arayüzle ilgili hataların giderildiğini **onaylamak amacıyla testler** yapılabilir.
- Sistem testleri için, belirli bir iş akışındaki **herhangi bir ekran değişirse**, bu iş akışına yönelik tüm testler yeniden koşturulabilir.
- Sistem entegrasyon testleri için, kredi puanlama mikroservisiyle etkileşen uygulamanın testleri, bu mikroservisin sürekli canlıya alınmasının bir parçası olarak **günlük bazda** tekrar koşturulabilir.
- Kabul testleri için, kabul testlerinde bulunan bir hata giderildikten sonra **daha önce başarısız olan tüm testler** tekrar koşturulabilir.

Bu bölümde, her seviyede her test çeşidinden örnekler sunulmuşsa da, tüm yazılımlar için her seviyede her test çeşidinin uygulanması gerekmeyebilir. Bununla birlikte, her seviyede, özellikle de test çeşidini ilgilendiren en erken seviyede, gerekli test çeşitlerini hayata geçirmek önemlidir.

2.4 Bakım Testleri(Maintenance Testing, Wartungstest)

***Üretim ortamlarına(production environments, Produktionsumgebungen)** alındıktan sonra yazılım ve sistemlerin bakımının yapılması gerekir. Teslim edilen yazılım ve sistemlerde, operasyonel kullanımda keşfedilen hataları gidermek, yeni fonksiyonalite eklemek veya önceden sunulan fonksiyonaliteyi kaldırmak veya değiştirmek için çeşitli değişiklikler yapılması neredeyse kaçınılmazdır. Birimin veya sistemin fonksiyonel olmayan kalite özelliklerini, özellikle performans, **uyumluluk(compatibility, Kompatibilität)**, güvenilirlik, güvenlik ve taşınabilirliği yazılımın kullanım ömrü boyunca korumak veya iyileştirmek için de bakım yapılması gerekir.*

Bakımın bir parçası olarak herhangi bir değişiklik yapıldığında, hem değişikliklerin ne kadar başarılı olduğunu değerlendirmek hem de sistemin değişmeyen kısımlarındaki (genellikle sistemin büyük

kısmıdır) olası yan etkileri (ör. regresyonlar) kontrol etmek için bakım testleri(Maintenance Testing, Wartungstest) yapılmalıdır.

Bakım testleri, sistemde yapılan değişiklikleri ve değişikliklerden etkilenmiş olabilecek değişmeyen parçaları test etmeye odaklanır. Bakım, planlanan ve planlanmamış sürümleri (düzeltmeler) içerebilir.

Bir bakım sürümü(maintenance release, Wartungsrelease), kapsamına bağlı olarak farklı test çeşitlerini kullanarak birçok test seviyesinde bakım testleri gerektirebilir.

Bakım testlerinin kapsamı(The scope of maintenance testing, Der Umfang von Wartungstests) aşağıdakilere bağlıdır:

- Değişikliğin risk derecesi, örneğin, yazılımın değiştirilen kısmının diğer birimlerle veya sistemlerle iletişim kurma derecesi
- Mevcut sistemin boyutu
- Değişikliğin boyutu

(Bakım testlerinin kapsamı(The scope of maintenance testing, Der Umfang von Wartungstests) değişikliğin risk derecesi, mevcut sistemin boyutu ve değişikliğin boyutuna bağlıdır)

2.4.1 Bakım için Tetikleyiciler(Triggers for Maintenance, Auslöser für Wartung)

Hem planlı hem de planlanmamış değişiklikler için yazılım bakımı ve dolayısıyla bakım testlerinin gerçekleştirilmesinin birçok nedeni vardır.

Bakım için tetikleyicileri(Triggers for Maintenance, Auslöser für Wartung) **aşağıdaki gibi sınıflandırabiliriz:**

- Değişiklik(Modification, Modifikation) : planlanan iyileştirmeler (örneğin sürüm bazlı), düzeltici ve acil durum değişiklikleri, operasyonel ortamdaki değişiklikler (planlanan işletim sistemi veya veritabanı yükseltmeleri gibi), ticari paket yazılımın (COTS) üst bir sürüme yükseltilmesi, hatalar ve güvenlik açıkları için yamalar gibi.
- Taşıma(Migration, Migration) : yeni ortamın ve değiştirilen yazılımın operasyonel testlerini gerektirebilen, bir platformdan diğerine taşıma gibi veya başka bir uygulamadan gelen verilerin dönüştürülme ihtiyacı sonucu testlerinin yapılması gibi.
- Kullanımdan kaldırma(Retirement, Außerbetriebnahme) : bir uygulamanın kullanım ömrünün sonuna gelinmesi gibi.
- Bir uygulama veya sistemin kullanımı sonlandırılırken, verilerin uzun süre saklanması gerekliyse, veri taşıma testleri(data migration, Datenmigration) veya arşivleme(archiving, Datenarchivierung) gerekebilir. Uzun saklama süreleri için veriler arşivlendikten sonra geri yükleme/geri alma prosedürlerinin(restore/retrieve procedures) test edilmesi de

gerekebilir. Ek olarak, kullanımda kalan herhangi bir fonksiyonalitenin hala çalıştığından emin olmak için **regresyon testleri gerekebilir.**

- Nesnelerin İnterneti için, **donanımlar ve yazılım servisleri gibi tamamen yeni veya değiştirilmiş öğelerin genel sisteme eklenmesi ile bakım testleri tetiklenebilir.** Bu tür sistemler için yapılan bakım testleri, farklı seviyelerde (örneğin, ağ seviyesi, uygulama seviyesi) entegrasyon testlerine ve özellikle kişisel verilere ilişkin güvenlik konularına özellikle önem vermektedir.

(Bakım için tetikleyiciler(Triggers for Maintenance, Auslöser für Wartung): Sistemde veya yazılımda bakım yapılmasını gerektiren etmenler: Değişiklik(Modification, Modifikation), Taşıma(Migration, Migration), Kullanımdan kaldırma(Retirement, Außerbetriebnahme), arşive kaldırılan ürünü geri yükleme/geri alma prosedürü(restore/retrieve procedures), Sisteme yeni Hardware veya Software eklenmesi)

2.4.2 Bakım için Etki Analizi(Impact Analysis for Maintenance, Auswirkungsanalyse für Wartung)

etki analizi(Impact analysis, Auswirkungsanalyse)**, bir değişikliğin beklenen ve olası yan etkilerini**, **istenilen sonuçların elde edilip edilemeyeceğini** ve **değişiklikten etkilenecek alanları belirlemek amacıyla yapılır**. Etki analizi bir değişikliğin mevcut testler üzerindeki etkisinin belirlenmesinde de yardımcı olabilir. Sistemdeki yan etkilerin ve etkilenen alanların regresyon testleri gerekli değişiklikler yapıldıktan sonra yapılmalıdır.

Bir değişiklik yapılmadan önce, sistemin diğer alanlarında yaratacağı potansiyel sonuçlara dayanarak değişikliğin yapılıp yapılmayacağına karar vermek için etki analizi(Impact analysis, Auswirkungsanalyse) yapılabilir.

Aşağıdaki durumlarda etki analizi zor olabilir:

- **Gereksinimler** (örneğin, iş gereksinimleri, kullanıcı hikâyeleri, mimari) **güncel değilse veya eksikse**
- **Test senaryoları dokümante edilmemişse veya güncel değilse**
- Testler ve test esası arasındaki çift yönlü izlenebilirlik korunmamışsa
- Araç desteği(Tool support, Werkzeugunterstützung) zayıfsa veya yoksa
- Katılan kişiler alan ve/veya sistem hakkında yeterli bilgiye sahip değilse
- Yazılım geliştirme sırasında yazılımın sürdürülebilirliğine yeterli özen gösterilmediyse

(Etki analizi(Impact analysis, Auswirkungsanalyse), bir değişikliğin beklenen ve olası yan etkilerini, istenilen sonuçların elde edilip edilemeyeceğini ve değişiklikten etkilenecek alanları belirlemek amacıyla yapılır. Analizin sonuçlarına göre değişiklik yapılıp yapılmamasına karar verilir. Gerekli değişikler yapıldıktan sonra regresyon testi yapılır.

Requirementlar eksik veya güncel değilse, Test case'ler yazılmamış veya güncel değilse, Testler ve Test base'ler arasında çift yönlü izlenebilirlik korunmamışsa, Tool support(tool desteği) zayıfsa veya yoksa, analize katılan kişiler uzman değilse, software sürecinde yeterli özen gösterilmediyse etki analizi yapmak zor olabilir)

CHAPTER3

3 Statik Testler

Anahtar kelimeler

kurgusuz gözden geçirme, kontrol listesine dayalı gözden geçirme, dinamik testler, resmi gözden geçirme, resmi olmayan gözden geçirme, teftiş, perspektife dayalı okuma, gözden geçirme, role dayalı gözden geçirme, senaryoya dayalı gözden geçirme, statik analiz, statik testler, teknik gözden geçirme, üzerinden geçme

Keywords
ad hoc review, checklist-based review, dynamic testing, formal review, informal review, inspection, perspective-based reading, review, role-based review, scenario-based review, static analysis, static testing, technical review, walkthrough

Schlüsselbegriffe
Ad-hoc-Review, checklistenbasiertes Review, dynamischer Test, formales Review, informelles Review, Inspektion, perspektivisches Lesen, Review, rollenbasiertes Review, statische Analyse, statischer Test, szenariobasiertes Review, technisches Review, Walkthrough

3.1 Statik Testin Temelleri

Test edilen yazılımın çalıştırılmasını gerektiren dinamik testlerin aksine **statik testler, yazılımın veya diğer çalışma ürünlerinin** manuel incelenmesine (gözden geçirmelere(reviews)) **veya kodun veya diğer çalışma ürünlerinin araç kullanılarak değerlendirilmesine (statik analizlere) dayanır**. Her iki statik test çeşidi de test edilen kod veya **çalışma ürününü gerçekten çalıştırmadan(without actually executing) değerlendirir**.

Statik analiz, güvenlik açısından kritik öneme sahip bilgisayar sistemleri (örneğin, havacılık, tıp veya nükleer yazılım) için önemlidir, ancak statik analiz diğer endüstrilerde ve test çeşitlerinde de önemli ve yaygın hale gelmiştir. Örneğin, **statik analiz güvenlik testlerinin önemli bir parçasıdır.** Statik analiz ayrıca çevik yazılım geliştirmede, sürekli teslim ve sürekli canlıya alımda, otomatik derleme ve teslim sistemlerine de dâhil edilir.

(statik testler, yazılımın veya diğer çalışma ürünlerinin manuel incelenmesine (gözden geçirmelere(reviews)) veya diğer çalışma ürünlerinin araç(tool) kullanılarak değerlendirilmesine (statik analizlere) dayanır. Çalışma ürünü(work product) gerçekten çalıştırmadan(without actually executing) değerlendirilir. Statik analiz güvenlik testlerinin önemli bir parçasıdır)

3.1.1 Statik Teste Dahil Edilebilecek Çalışma Ürünleri

Her türlü çalışma ürünü statik testler (gözden geçirme ve/veya statik analiz) kullanılarak incelenebilir, örneğin:

- İş gereksinimleri(business requirement, Fachanforderungen), **fonksiyonel gereksinimler ve güvenlik gereksinimleri** dâhil olmak üzere tüm gereksinim çeşitleri
- Epikler, kullanıcı hikâyeleri ve kabul kriterleri
- Mimari ve tasarım gereksinimleri
- Kod
- Test planları, test senaryoları, test prosedürleri ve otomatikleştirilmiş test betikleri dâhil olmak üzere test için kullanılan yazılımlar
- Kullanıcı kılavuzları
- Web sayfaları
- Sözleşmeler, proje planları, zaman çizelgeleri ve bütçeleri
- Model Bazlı testler için kullanılabilecek aktivite diyagramları gibi modeller (bkz. ISTQB-MBT Temel Seviye Model Bazlı Test Uzmanı Devam Kursu Ders Programı ve Kramer 2016)

Gözden geçirmeler(Reviews), katılımcıların okuyup anlayabileceği her çalışma ürününe uygulanabilir. **Statik analiz**, üzerinde kullanılabilecek uygun bir statik analiz aracının bulunduğu **düzenli bir yapıya sahip her çalışma ürününe (genellikle kod veya modeller) verimli bir şekilde uygulanabilir**. Statik analiz, gereksinimler gibi doğal dilde yazılmış çalışma ürünlerine (örneğin yazım, dilbilgisi ve okunabilirlik denetimi gibi) bile uygulanabilir.

(Gözden geçirmeler(Reviews), katılımcıların okuyup anlayabileceği her çalışma ürününe uygulanabilir. Statik analiz, düzenli bir yapıya sahip her çalışma ürününe (genellikle kod veya modeller) verimli bir şekilde uygulanabilir)

3.1.2 Statik Testin Faydaları

Statik test teknikleri çeşitli faydalar sağlar. Yazılım geliştirme yaşam döngüsünün erken dönemlerinde uygulandığında **statik testler, dinamik testler yapılmadan önce hataların erkenden bulunmasına olanak sağlar** (örneğin, gereksinimlerin veya tasarımın gözden geçirmelerinde, ürün iş listesinin iyileştirilmesinde vb). **Erken bulunan hataların düzeltilmesi**, özellikle yazılım canlıya alındıktan sonra veya yazılım yaşam döngüsünün ileri aşamalarında bulunan hatalarla karşılaştırıldığında **çok daha düşük maliyetlidir**. Özellikle hata ile ilgili çalışma ürünlerinin güncellenmesi, onaylama ve regresyon testlerinin yapılması için gereken ek maliyetler göz önüne alındığında, hataları bulmak için statik test tekniklerini kullanmak ve bu hataları hemen düzeltmek, test nesnesindeki hataları bulmak için dinamik testleri kullanmaktan ve sonrasında bunları düzeltmekten neredeyse her zaman daha az maliyetlidir.

(statik testler, dinamik testler yapılmadan önce hataların erkenden bulunmasına olanak sağlar Erken bulunan hataların düzeltilmesi çok daha düşük maliyetlidir)

Statik testlerin ek faydaları arasında aşağıdakiler sayılabilir:

- **Hataların daha verimli bir şekilde**, **dinamik testlerin yürütülmesinden önce bulunması** ve **düzeltilmesi**
- **Dinamik testlerle kolayca bulunamayan hataların bulunması**
- Gereksinimlerdeki tutarsızlıkları, belirsizlikleri, çelişkileri, çıkarmaları, **yanlışlıkları ve fazlalıkları ortaya çıkararak tasarımdaki(design) veya kodlamadaki(coding) hataların önlenmesi**
- Yazılım geliştirme verimliliğinin artırılması (ör. gelişmiş tasarım ve daha sürdürülebilir kod sayesinde)
- Yazılım geliştirme maliyet ve süresinin düşürülmesi
- Testlerin **maliyet ve süresinin düşürülmesi**
- Yaşam döngüsünün ilerleyen kısımlarında veya canlıya alımdan sonra daha az arıza olması sebebiyle yazılımın kullanım ömrü boyunca gerçekleşecek olan toplam kalite maliyetinin düşürülmesi
- **Gözden geçirmeler sırasında ekip üyeleri arasındaki iletişimin iyileştirilmesi**

(Dinamik testte hataların bulunması, Statik testte hataların önlenmesi sözkonusudur. Bu yönüyle Statik test daha verimlidir. Maliyetin düşürülmesine, zaman kaybedilmesini engellemeye, yaşam döngüsünün ilerleyen kısımlarında daha az arıza olmasına, bu sayede yazılımın kullanım ömrü boyunca bakım maliyetlerinin düşmesine ve toplam kalite maliyetinin düşürülmesine, review sırasında ekip üyeleri arasındaki iletişimin iyileştirilmesine katkı sağlama gibi faydaları vardır)

3.1.3 Statik ve Dinamik Testler Arasındaki Farklar

Statik testler ve dinamik testler aynı hedeflere sahip olabilir (bkz. Bölüm 1.1.1); ör. **çalışma ürünlerinin kalitesinin değerlendirilmesini sağlamak ve hataları mümkün olan en erken zamanda bulmak** gibi. **Statik ve dinamik testler, farklı türdeki hataları bularak birbirini tamamlarlar.**

Önemli farklarından birisi de **statik testlerin**, yazılım çalışırken hataların neden olduğu arızaları bulmak yerine doğrudan **çalışma ürünlerindeki hataları çalıştırmadan bulmasıdır**. Bir hata, çalışma ürününde arızaya neden olmadan çok uzun süre kalabilir. Hatanın bulunduğu senaryo nadiren denenen veya koşturulması zor bir senaryo olabilir, bu nedenle bu hatayla karşılaşacak dinamik bir test oluşturmak ve koşturmak kolay olmayacaktır. **Statik testler hatayı çok daha az eforla bulabilir**.

Diğer bir ayrım ise, **statik testlerin çalışma ürünlerinin tutarlılığını** ve **iç kalitesini iyileştirmek için kullanılması**, **dinamik testlerin** ise genellikle dışarıdan görülebilen **davranışlara odaklanmasıdır**.

(Statik ve dinamik testler, farklı türdeki hataları bularak birbirini tamamlarlar. Çalışma ürünlerinin(work products) kalitesinin değerlendirilmesini sağlamak ve hataları mümkün olan en erken zamanda bulmak gibi ortak hedefleri vardır.
Statik testlerde, work producttaki hatalar kodlar çalıştırılmadan bulunur.
Dinamik testle uzunca bir süre uğraşıp test yazarak bulunabilecek bir hata, Statik testle çok daha az bir eforla bulunabilir.
Statik testler work productların tutarlılığını ve iç kalitesini iyileştirmek için kullanılır. Dinamik testler ise work productların iç yapısını görmeden sadece davranışlarına odaklanır)

Dinamik testlerle karşılaştırıldığında, **statik testler ile bulunması ve düzeltilmesi daha kolay ve daha ucuz olan yaygın hatalar** şunlardır:

- Gereksinim hataları (örneğin tutarsızlıklar, belirsizlikler, çelişkiler, çıkarmalar, yanlışlıklar ve fazlalıklar)
- Tasarım hataları (örneğin verimsiz algoritmalar veya veritabanı yapıları, yüksek bağlaşım, düşük uyum)
- Kodlama hataları (örneğin, değer atanmamış değişkenler, tanımlanmış fakat hiç kullanılmamış değişkenler, ulaşılamayan kod, tekrarlanan kod)
- Standartlardan sapmalar (örneğin, kodlama standartlarına bağlı kalmama)
- Hatalı arayüz gereksinimleri (örneğin, çağıran sistem ile çağrılan sistem tarafından kullanılan ölçüm birimlerinin farklı olması)
- Güvenlik açıkları (örneğin, arabellek aşımlarına karşı duyarlılık)
- Test esası izlenebilirliğinde veya kapsamında boşluklar veya yanlışlıklar (örneğin bir kabul kriteri için testlerin eksik olması)
- Ayrıca, çoğu sürdürülebilirlik hatası(maintainability defects, Wartbarkeitsfehler) **sadece statik testler** (örneğin, hatalı modülerleştirme, birimlerin tekrar kullanılabilirliğinin düşük olması, analiz edilmesi ve değiştirilmesi zor olan kodlar) **ile bulunabilir.**

(statik testler ile bulunması ve düzeltilmesi daha kolay ve daha ucuz olan yaygın hatalar; Gereksinim hataları, Tasarım hataları, Kodlama hataları, Standartlardan sapmalar, Hatalı arayüz gereksinimleri, Güvenlik açıkları, Test basedeki eksikler gibi hatalar. Ayrıca, çoğu sürdürülebilirlik hatası(maintainability defects, Wartbarkeitsfehler) sadece statik testler ile bulunabilir)

3.2 Gözden Geçirme Süreci(Review Process, Reviewprozess)

Gözden geçirmeler, resmi veya gayri resmi olabilir. Gayri resmi gözden geçirmeler(Informal reviews, Informelle Reviews) **tanımlanmış bir süreci takip etmez ve resmi olarak dokümante edilen çıktıları yoktur.** Resmi gözden geçirmelerin(Formal reviews) özellikleri arasında ise **ekip katılımı**, **gözden geçirme sonuçlarının dokümante edilmesi** ve gözden geçirmenin işletilmesi için dokümante edilmiş prosedürler yer alır. **Gözden geçirme sürecinin resmiliği; yazılım geliştirme yaşam döngüsü modeli**, **yazılım geliştirme sürecinin olgunluğu**, gözden geçirilecek **çalışma ürününün karmaşıklığı**, varsa **yasal veya düzenleyici gereksinimler** ve/veya **bir denetim izlemesi gerekliliği** gibi faktörlerle ilgilidir.

Gözden geçirmenin odağı, gözden geçirmenin kararlaştırılan hedeflerine (örneğin; hataları bulmak, anlamak, test uzmanları ve yeni ekip üyeleri gibi katılımcıları eğitmek veya fikir birliği ile görüşüp karara bağlamak) **bağlıdır.**

(Gayri resmi gözden geçirmeler(Informal reviews, Informelle Reviews) tanımlanmış bir süreci takip etmez ve resmi olarak dokümante edilen çıktıları yoktur.

Resmi gözden geçirmelerin(Formal reviews) özellikleri arasında ekip katılımı, sonuçların dokümante edilmesi ve bunların prosedürleri yer alır.

Gözden geçirme sürecinin resmiliği; yazılım geliştirme yaşam döngüsü modeli, yazılım geliştirme sürecinin olgunluğu, gözden geçirilecek çalışma ürününün karmaşıklığı, varsa yasal veya düzenleyici gereksinimler ve bir denetim izlemesi gerekliliğine bağlıdır.

Review hedeflerine bağlı olarak odak noktası değişir. (örneğin; hataları bulmak, anlamak, test uzmanları ve yeni ekip üyeleri gibi katılımcıları eğitmek veya fikir birliği ile görüşüp karara bağlamak))

3.2.1 Çalışma Ürünü Gözden Geçirme Süreci(Work Product Review Process, Reviewprozess für Arbeitsergebnisse)

Gözden geçirme süreci aşağıdaki ana aktivitelerden oluşur:

Planlama

- **Kapsamın tanımlanması:** gözden geçirm**enin amacını**, **hangi dokümanların** veya dokümanların hangi bölümlerinin **gözden geçirileceğini ve değerlendirilecek kalite özelliklerini içerir**
- Efor ve zamanın tahmin edilmesi
- Rollerle birlikte gözden geçirme **çeşidi, aktiviteler ve kontrol listeleri** gibi gözden geçirme özelliklerinin belirlenmesi
- Gözden geçirmeye **katılacak kişilerin seçilmesi ve rol paylaşımı**
- Daha resmi gözden geçirme türleri için (örneğin, teftişler) **giriş ve çıkış kriterlerinin tanımlanması**
- **Giriş kriterlerinin karşılandığının kontrol edilmesi** (daha resmi gözden geçirme türleri için)

Gözden geçirmenin başlatılması

- Çalışma ürünü, bulgu kayıt formları, kontrol listeleri ve ilgili çalışma ürünleri gibi diğer **materyallerin dağıtılması** (fiziksel olarak veya elektronik ortamda)
- **Katılımcılara kapsamın, hedeflerin, sürecin, rollerin ve çalışma ürünlerinin açıklanması**
- Katılımcıların varsa gözden geçirme ile ilgili **sorularının cevaplanması**

Bireysel gözden geçirme (bireysel hazırlanma)

- **Çalışma ürününün tamamının veya bir kısmının gözden geçirilmesi**
- **Potansiyel hataların, bulguların, önerilerin ve soruların not alınması**

Bulguların iletilmesi ve analizi

- Belirlenen olası hataların iletilmesi (örneğin bir gözden geçirme toplantısında)
- **Potansiyel hataların analiz edilmesi,** bunlar için bir sorumlu ve durumun atanması
- **Kalite özelliklerinin değerlendirilmesi ve dokümante edilmesi**

- **Gözden geçirme kararı** (reddedildi; büyük değişiklikler gerekli; olası küçük değişikliklerle kabul edildi) vermek için gözden geçirme bulgularının **çıkış kriterlerine göre değerlendirilmesi**

Hataların giderilmesi ve raporlama

- **Değişiklik gerektiren bulgular için hata raporlarının oluşturulması**
- Gözden geçirilen çalışma ürününde tespit edilen **hataların çözülmesi** (genellikle çalışma ürününün yazarı/oluşturucusu tarafından yapılan)
- **Hataların** (gözden geçirilen çalışma ürünüyle bağlantılı başka bir çalışma ürününde bulunduğunda) **uygun kişi veya ekibe iletilmesi**
- Olanaklar dâhilinde yorumu yapan kişinin onayını da alarak, güncellenmiş hata durumlarının (resmi gözden geçirmelerde) kaydedilmesi
- **Metriklerin toplanması** (resmi gözden geçirme türleri için)
- **Çıkış kriterlerinin karşılandığının kontrol edilmesi** (resmi gözden geçirme türleri için)
- **Çıkış kriterleri sağlandığında çalışma ürününün kabul edilmesi**

Bir çalışma ürünü gözden geçirmesinin sonuçları review türüne ve resmi(formel) olup olmamasına bağlı olarak değişir.

3.2.2 Resmi Gözden Geçirmede Roller ve Sorumluluklar

Tipik bir resmi gözden geçirme aşağıdaki rolleri içerir:

Yazar (çalışma ürünü yazarı/oluşturucu)(Author)

- Gözden geçirilen **çalışma ürününü oluşturur**
- Gözden geçirilen çalışma ürünündeki **hataları düzeltir**(gerekirse)

Author : work product'ı oluşturur, hataları düzeltir.

Yönetim(Management)

- Gözden geçirme **planlamasından sorumludur**
- Gözden geçirmelerin **uygulanmasına karar verir**
- **Personel, bütçe ve zaman tahsis eder**
- Mevcut maliyet etkinliğini izler
- Sürecin verimli işletilmesiyle ilgili kararları verir

Review'in planlamasından sorumludur, uygulanmasına karar verir. Personel, bütçe ve zaman tahsis eder.

Moderatör(Facilitator (often called moderator), Reviewmoderator)

- Gözden geçirme toplantılarının etkin bir şekilde yürütülmesini sağlar
- Gerekirse çeşitli bakış açıları arasında arabuluculuk yapar
- Genellikle gözden geçirmenin başarısının bağlı olduğu kişidir

Toplantıların etkin bir şekilde yürütülmesini sağlar, ekip arasında arabuluculuk yapar, review'in başarısı Moderatörün performansına bağlıdır.

Gözden geçirme lideri(Review leader, Reviewleiter)

- Gözden geçirmenin genel sorumluluğunu üstlenir
- Sürece kimlerin dâhil olacağına karar verir ve ne zaman ve nerede gerçekleştirileceğini organize eder

Review nerede, ne zaman yapılacak organizeyi gerçekleştirir, review'in genel sorumlusudur.

Gözden geçiriciler(Reviewers, Gutachter)

- Konunun uzmanları, projede çalışan kişiler, çalışma ürünü üzerinde söz sahibi olan paydaşlar ve/veya belirli teknik veya iş tecrübesine sahip kişiler olabilir
- Çalışma ürünündeki bulguları bulurlar
- Farklı bakış açılarını temsil edebilir (örneğin, test uzmanı, programcı, kullanıcı, operatör, iş analisti, kullanılabilirlik uzmanı, vb.)

Çalışılan ürün üzerinde Teknik veya business tecrubesi ve bilgi sahibi olan, gözden geçirme işlemini yapan kişiler.

Yazıcı (veya kaydedici)(Scribe (or recorder), Protokollant)

- Her bir gözden geçirme faaliyeti sırasında bulunan bulguları bir araya getirir ve sıraya koyar
- Gözden geçirme toplantısında belirlenen yeni potansiyel hataları, açık noktaları ve kararları kaydeder

Bazı gözden geçirme çeşitlerinde bir kişi birden fazla rol oynayabilir ve her rolle ilgili görevler gözden geçirme çeşidine göre değişiklik gösterebilir. Ek olarak, gözden geçirme sürecini, özellikle hataların, açık noktaların ve kararların kaydedilmesini destekleyen araçların ortaya çıkmasıyla, çoğu zaman bir yazıcıya gerek duyulmaz.

3.2.3 Gözden Geçirme Çeşitleri(Review Types, Reviewarten)

Gözden geçirmeler çeşitli amaçlar için kullanılabilse de temel amaçlarından biri hataları bulmaktır. Tüm gözden geçirme çeşitleri hata bulmaya yardımcı olabilir; **gözden geçirme türünün seçimi**, diğer seçim kriterlerinin yanı sıra, **projenin ihtiyaçlarına, mevcut kaynaklara, ürün çeşidine ve risklerine, iş alanına ve şirket kültürüne dayanmalıdır.**

Gözden geçirmeler çeşitli özelliklere göre sınıflandırılabilir. Aşağıda en yaygın dört gözden geçirme çeşidi ve özellikleri listelenmiştir.

(Çok farklı amaçlar için review yapılsada en önemli amaç hataların ortaya çıkarılmasıdır(main objectives).
Review türünün seçimi, projenin ihtiyaçlarına, mevcut kaynaklara, ürün çeşidine ve risklerine, iş alanına ve şirket kültürüne göre değişir.
En yaygın dört gözden geçirme çeşidi informel'den very formel'e doğru özellikleri ile aşağıda listelenmiştir)

Gayri resmi gözden geçirme(Informal review) (örneğin; çalışma arkadaşının kontrol etmesi, eşleştirme, eşli gözden geçirme)

- Ana amaç: **potansiyel hataların bulunması.**
- Diğer amaçlar: **yeni fikirler veya çözümler üretmek, küçük problemleri hızla çözmek.**
- **Resmi (dokümante edilmiş) bir sürece dayanmaz.**
- Bir gözden geçirme toplantısı **düzenlenmeyebilir**.
- **Yazarın meslektaşı (çalışma arkadaşı kontrolü) veya daha fazla kişi tarafından gerçekleştirilebilir**
- Sonuçlar **dokümante edilebilir.**
- Gözden geçiricilere bağlı olarak faydası değişir.
- **Kontrol listelerinin kullanımı isteğe bağlıdır.**
- **Çevik yazılım geliştirmede çok yaygın kullanılır.**

Üzerinden geçme(Walkthrough)

- Ana amaçlar: **hataların bulunması, yazılımın iyileştirilmesi, alternatif uygulamaların dikkate alınması, standartlara ve gereksinimlere uygunluğun değerlendirilmesi**.
- Diğer amaçlar: **teknikler veya yöntem farklılıkları hakkında fikir alışverişinde bulunmak, katılımcıların eğitimi, fikir birliğine varmak**.
- Gözden geçirme toplantısından önce bireysel hazırlık **isteğe bağlıdır**.
- Gözden geçirme toplantısı genellikle çalışma ürününün **yazarı tarafından yönetilir**.
- Yazıcının olması **zorunludur.**
- Kontrol listelerinin kullanımı **isteğe bağlıdır**.
- **Senaryolar, provalar veya simülasyonlar şeklinde olabilir**.
- **Potansiyel hata kayıtları ve gözden geçirme raporları oluşturulabilir**.
- Uygulamada gayrı resmiden, çok resmi arasında değişiklik gösterebilir.

Teknik gözden geçirme(Technical review)

- Ana amaçlar: **fikir birliğine varmak, potansiyel hataların bulunması**.
- Diğer amaçlar: **kalitenin değerlendirilmesi ve çalışma ürünü için güven oluşturulması, yeni fikirler üretilmesi, alternatif çözümleri göz önünde bulundurarak gelecekteki iş ürünlerini iyileştirmek için paydaşları motive etmek ve olanak sağlamak**.
- Gözden geçiriciler **yazarın teknik olarak dengi olan diğer veya aynı disiplinlerdeki teknik uzmanlardan seçilmelidir**.
- Gözden geçirme toplantısından önce **bireysel hazırlık gereklidir**.
- Gözden geçirme toplantısı **isteğe bağlı olup, ideal olarak eğitimli bir moderatör (genellikle çalışma ürününün geliştirilmesine dâhil olmayan kişiler) tarafından yönlendirilir**.
- Yazıcı **zorunludur, idealde çalışma ürünü yazarından farklı biri olmalıdır**.
- **Kontrol listelerinin kullanımı isteğe bağlıdır**.
- **Potansiyel hata kayıtları ve gözden geçirme raporları genellikle oluşturulur**.

Teftiş(Inspection)

- Ana amaçlar: **Potansiyel hataların bulunması, çalışma ürünündeki kaliteyi değerlendirmek ve güven oluşturmak, çalışma ürününü yazanların öğrenmesi ve kök neden analizi yoluyla gelecekteki benzer hataları önlemek.**
- Diğer amaçlar: **ürün sahiplerini gelecekteki çalışma ürünlerini ve yazılım geliştirme sürecini iyileştirmeye motive etmek ve olanak sağlamak, fikir birliği sağlamak**.
- **Kurallara ve kontrol listelerine dayanarak, resmi, dokümante çıktılarla tanımlanmış bir süreci izler**.
- Bölüm 3.2.2'de belirtilenler gibi ve **zorunlu olan, açıkça tanımlanmış roller kullanır ve (gözden geçirme toplantısında çalışma ürününü yüksek sesle okuyacak) özel bir okuyucu yer alabilir**.
- Gözden geçirme toplantısından önce **bireysel hazırlık gereklidir**.
- **Gözden geçiriciler çalışma ürünü yazarının dengi veya çalışma ürünü ile ilgili diğer disiplinlerde uzman kişilerdir.**
- **Belirlenen giriş ve çıkış kriterleri kullanılır**.
- **Yazıcı olması zorunludur**.
- Gözden geçirme toplantısı **eğitimli bir moderatör (çalışma ürünü yazarı dışında) tarafından yönetilir**.
- **Yazar; gözden geçirme lideri, okuyucu veya yazıcı olarak görev yapamaz**.
- **Potansiyel hata kayıtları ve gözden geçirme raporu oluşturulur**.
- **Metrikler toplanır ve teftiş süreci de dâhil olmak üzere tüm yazılım geliştirme sürecini iyileştirmek için kullanılır**.

Tek bir çalışma ürününe, birden fazla gözden geçirme çeşidi uygulanabilir. Birden fazla gözden geçirme çeşidi kullanılıyorsa, uygulama sıralaması değişebilir. Örneğin, çalışma ürününün teknik gözden geçirmeye hazır olduğundan emin olmak için teknik gözden geçirmeden önce gayri resmi bir gözden geçirme yapılabilir.

Yukarıda açıklanan gözden geçirme çeşitleri eş-gözden geçirme olarak benzer organizasyonel seviyedeki meslektaşlar tarafından yapılabilir.

Gözden geçirme sürecinde bulunan hata çeşitleri, özellikle gözden geçirilen çalışma ürününe bağlı olarak değişiklik gösterir. Farklı çalışma ürünlerindeki gözden geçirmelerde bulunabilecek hata örnekleri için bölüm 3.1.3'e bakınız ve teftişler hakkında daha fazla bilgi almak için Gilb 1993'e bakınız.

EXCEL REVİEW ÇEŞİTLERİNİ İNCELE

3.2.4 Gözden Geçirme Tekniklerinin Uygulanması(Applying Review Techniques, Die Anwendung von Reviewverfahren)

Hataları bulmak için bireysel gözden geçirme (bireysel hazırlık) faaliyeti sırasında uygulanabilecek birkaç gözden geçirme tekniği vardır. **Bu teknikler yukarıda açıklanan gözden geçirme çeşitlerinde kullanılabilir. Tekniklerin etkinliği** kullanılan **gözden geçirme çeşidine bağlı olarak değişebilir.** Çeşitli gözden geçirme türleri için farklı bireysel gözden geçirme tekniklerinin örnekleri aşağıda listelenmiştir:

(Yukarıda bahsedilen 4 Review çeşidine aşağıdaki teknikler uygulanabilir. Bu tekniklerin etkinliği Review çeşidine göre değişir)

Kurgusuz(Ad hoc)

Kurgusuz gözden geçirmede(Ad-hoc-Review) gözden geçiricilere **bu görevin nasıl gerçekleştirilmesi gerektiği hakkında çok az rehberlik sağlanır veya hiç sağlanmaz.** Gözden geçiriciler genellikle çalışma **ürününü sırayla okur ve karşılaştıkları sorunları dokümante eder.**

Kurgusuz gözden geçirme **çok az hazırlık gerektiren ve yaygın kullanılan bir tekniktir.** Bu teknik büyük ölçüde **gözden geçiricilerin becerilerine bağlıdır** ve **farklı gözden geçiriciler tarafından bildirilen birçok benzer sorunun bulunmasına yol açabilir.**

(Work product sırayla okunur ve karşılaşılan sorunlar dokümante edilir. Herhangi bir rehberlik sağlanmadan, herhangi bir hazırlığa ihtiyaç duymadan yaygın olarak kullanılan bir tekniktir. Farklı reviewerlar tarafından tekrar sayılabilecek benzer hataların bulunması bir dezavantajdır)

Kontrol listesine dayalı(Checklist-based, Checklistenbasiert)

Kontrol listesine dayalı gözden geçirme (Checklist-based review, Checklistenbasiert review)
gözden geçiricilerin gözden geçirme başlangıcında (örneğin **moderatör tarafından) dağıtılan kontrol listelerine dayanarak sorunları belirlediği sistematik bir tekniktir.** Gözden geçirme **kontrol listesi deneyimle elde edilen ve olası hatalara dayanan bir dizi sorudan oluşur**. **Kontrol listeleri** gözden geçirilmekte olan çalışma ürününe özel olmalı ve önceki gözden geçirmelerde kaçırılan sorun türlerini kapsayacak şekilde düzenli olarak **güncellenmelidir.**

Kontrol listesine dayalı tekniğin **en önemli avantajı tipik hata çeşitlerinin sistematik bir şekilde kapsanmasıdır.** Bireysel gözden geçirmelerde **sadece kontrol listesini takip etmekle yetinmemeye ve kontrol listesinin dışındaki hataları da aramaya özen gösterilmelidir.**

(Moderatör tarafından dağıtılan tecrübeye dayalı kontrol listeleri kullanılarak yapılır. Sadece kontrol listesi ile sınırlı değildir, bu liste dışındaki hatalarda aranır. Kontrol listesi liste dışı hatalar bulundukça güncellenerek sonraki reviewler için önemli bir tecrübe birikimi sağlanır)

Senaryolar ve provalar(Scenarios and dry runs, Szenarien und Dry Runs (Probeläufe))

Senaryoya dayalı gözden geçirmede (Scenarios and dry runs review, Szenarien und Dry Runs (Probeläufe review))
gözden geçiricilere çalışma ürününü nasıl okuyacaklarına ilişkin rehberler sağlanır. Senaryoya dayalı yaklaşım, (çalışma ürünü, kullanım senaryoları gibi uygun bir biçimde dokümante edilmişse) **çalışma ürünü üzerinde "provalar" yapılmasını sağlayarak gözden geçiricilerin işini kolaylaştırır.** Bu senaryolar, gözden geçiricilere hata çeşitlerini bulma konusunda **basit kontrol listelerinden daha iyi rehberlik sağlar.**

Kontrol listesine dayalı gözden geçirmelerde olduğu gibi diğer hata çeşitlerini (örneğin, eksik özellikler) kaçırmamak için, gözden geçiricilerin **dokümante senaryolarla sınırlandırılmaması gerekir.**

(Reviewer'lara work product üzerinde prova yapmalarını sağlayacak, kontrol listelerinden daha verimli bir kullanım senaryosu verilerek rehberlik sağlanır. Reviewer'lar bu senaryodan bağımsız olarak ta hata aramalıdır)

Role dayalı(Role-based, Rollenbasiert)

Role dayalı gözden geçirme (Role-based review, Rollenbasiert Review)
, gözden geçiricilerin çalışma ürününden etkilenen **her bir paydaşın(stakeholder) rolleri gözünden çalışma ürününü değerlendirdiği bir tekniktir.** Genellikle roller arasında farklı son kullanıcı grupları (deneyimli, deneyimsiz, kıdemli, çocuk vb.) ve kurumdaki belirli roller (kullanıcı yöneticisi, sistem yöneticisi, performans testi uzmanı vb.) yer alır.

(End-userların kim olduğuna bağlı olarak yapılan Review tekniğidir.
Ürünü kullanacak kullanıcı çeşidi(roller) dikkate alınarak yapılan Review tekniğidir. (Çocuk, deneyimli, deneyimsiz, yönetici, performans testi uzmanı, sistem yöneticisi)

Bakış açısına dayalı(Perspective-based, Perspektivisch)

Bakış açısına dayalı okumada (Perspective-based, Perspektivisch)
, role dayalı gözden geçirmeye benzer şekilde, gözden geçiriciler bireysel gözden geçirme aşamasında **farklı paydaş(stakeholder) bakış açılarını dikkate alır.** Tipik paydaş bakış açıları arasında son kullanıcı, pazarlama, tasarımcı, test uzmanı veya kurum yönetimi yer alır. **Farklı paydaş bakış açılarının kullanılması bireysel gözden geçirmede daha fazla derinliğe ulaşılmasını sağlar ve gözden geçiriciler tarafından benzer sorunları bulunma olasılığını azaltır.**

Bakış açısına dayalı okuma ayrıca gözden geçiricilerin, çalışma ürününü kullanmaya çalışmasını da gerektirir. Örneğin bir test uzmanı, **gereksinimler üzerinde son kullanıcının bakış açısına dayalı bir okuma yapıyorsa**, gerekli tüm bilgilerin dâhil edilip edilmediğini görmek için taslak kabul testleri(draft acceptance tests) **oluşturmaya çalışacaktır**. Ayrıca, **bakış açısına dayalı okumalarda kontrol listelerinin kullanılması beklenir**.

Yapılan çalışmalar, bakış açısına dayalı okumanın **gereksinimleri ve teknik çalışma ürünlerini gözden geçirmek için en etkili teknik olduğunu göstermiştir**. Farklı paydaş bakış açılarını sürece dâhil etmek ve risklere dayanarak uygun şekilde ağırlıklarını belirlemek önemli başarı faktörlerinden biridir. Bakış açısına dayalı okuma hakkında ayrıntılı bilgi için Shul 2000'e ve farklı gözden geçirme türlerinin etkinliği için Sauer 2000'e bakınız.

(Review esnasında farklı paydaş(Stakeholder) bakış açılarını dikkate alır(marketing, designer, tester, operations) pazarlama, tasarımcı, test uzmanı veya kurum yönetimi gibi düşünerek gözden geçirmeleri yapar. Farklı paydaş bakış açılarının kullanılması daha iyi neticelere ulaşılmasına ve reviewerlar tarafından benzer sorunların bulunma olasılığını azaltır.

Bir son kullanıcı gibi düşünüp Taslak kabul testleri hazırlanır. Ayrıca bu review tekniğinde kontrol listelerinin kullanılması beklenir.
Gereksinimleri ve teknik çalışma ürünlerini gözden geçirmek için kullanılan en etkili tekniktir)

3.2.5 Gözden Geçirmelerin Başarı Faktörleri(Success Factors for Reviews, Erfolgsfaktoren für Reviews)

Başarılı bir gözden geçirme gerçekleştirmek için **uygun gözden geçirme çeşidi ve tekniği kullanılmalıdır**. Bunun yanında, gözden geçirmenin sonucunu etkileyecek bir dizi başka faktör vardır.

Gözden geçirmenin organizasyonel başarı faktörleri (Organizational success factor, Organisatorische Erfolgsfaktoren) aşağıdaki gibidir:

- Her gözden geçirme, gözden geçirme planlaması sırasında tanımlanan ve ölçülebilir **çıkış kriterleri olarak kullanılan net hedeflere sahiptir**
- Hedeflere ulaşmak için uygun olan ve gözden geçirilecek çalışma ürünlerinin **türüne** ve **seviyesine** ayrıca **katılımcılara** uygun gözden geçirme çeşitleri uygulanır
- **Gözden geçirme tekniklerinin tümü**, kontrol listesine dayalı veya role dayalı gözden geçirme gibi, gözden geçirilecek çalışma ürünündeki **hataların etkili bir şekilde bulunmasına yardımcı olur**
- Kullanılan tüm **kontrol listeleri ana riskleri ele alır ve günceldir**
- **Büyük dokümanlar küçük parçalar halinde ele alınır** ve gözden geçirilir, böylece dokümanı yazanlara hatalar hakkında erken ve sık geri bildirimde bulunularak kalite kontrolü sağlanır
- Katılımcıların hazırlanmak için **yeterli zamanı vardır**
- Gözden geçirmeler, uygun şekilde **önceden bildirimle planlanır**
- **Yönetim, gözden geçirme sürecini destekler** (örneğin proje planlarına gözden geçirme faaliyetleri için yeterli sürenin eklenmesiyle)

(Çıkış kriterleri olarak kullanılan net hedeflere sahiptir. Çalışma ürünün türüne, seviyesine ve katılımcılara göre review çeşidi uygulanır. Kontrol listeleri ana riskleri ele alır ve günceldir. Büyük dökümanlar küçük parçalar halinde ele alınır. Review için yeterli zaman tanınır. Önceden bildirimle planlanır. Yönetim Review sürecini destekler, projenin planlamasında buna yer ayırır)
Gözden geçirmelerin kişilere bağlı başarı faktörleri (People-related success factors, Personenbezogene Erfolgsfaktoren) aşağıdaki gibidir:

- Gözden geçirme hedeflerini yerine getirmek için **doğru kişiler** sürece dâhil olur; örneğin, dokümanı çalışmalarında kullanabilecek farklı beceri gruplarına veya bakış açılarına sahip kişiler
- **Test uzmanları**, gözden geçirmeye katkıda bulunan ve bu sayede çalışma ürünü hakkında bilgi sahibi olan değerli gözden geçiriciler olarak görülür, bu da onların daha etkin testler hazırlamasına ve bu testleri erkenden hazırlamasına olanak sağlar
- Katılımcılar **detaylara yeterince zaman ayırır ve özen gösterir**
- **Gözden geçirmeler küçük parçalar üzerinde yapılır**, böylece gözden geçiriciler bireysel gözden geçirme ve/veya (yapılıyorsa) gözden geçirme toplantısı sırasında konsantrasyonlarını kaybetmezler
- **Belirlenen hatalar olumlu bir şekilde karşılanır**, takdir edilir ve nesnel olarak değerlendirilir

- **Toplantı iyi yönetilir**, böylece katılımcılar bunu zamanlarının faydalı bir şekilde kullanımı olarak görürler
- Gözden geçirme bir **güven ortamında** gerçekleştirilir; çıktı, **katılımcıların kişisel olarak değerlendirilmesinde kullanılmaz**
- Katılımcılar; sıkılma, öfke veya diğer katılımcılara düşmanlık olarak anlaşılabilecek beden dilinden ve davranışlardan kaçınırlar
- Özellikle **teftişler gibi daha resmi gözden geçirme çeşitleri için yeterli eğitim sağlanır**
- Öğrenme ve süreç iyileştirme kültürü desteklenir

(Review'de görev alan kişiler; doğru kişilerden seçilir, Test uzmanları yer alır, detaylara yeterince zaman ayrılır ve özen gösterilir, gözden geçirmeler küçük parçalar üzerinde yapılır, hata bulan kişiler takdir edilir, olumlu şekilde karşılanır, toplantılar iyi yönetilir, şirkette şahısların çalışmaları kişilerin değerlendirilmesinde kullanılmayacağı güveni verilir, arkadaşlık ortamı oluşturulur, teftiş(Inspection) gibi review çeşitleri için gerekli eğitimler verilir, öğrenme ve süreç iyileştirme kültürü desteklenirse reviewler bu faktörlere riayet edilmesi ölçüsünde başarılı olur)

CHAPTER 4

4. TEST TASARIM TEKNİKLERİ

Anahtar kelimeler

kara kutu test tekniği, sınır değer analizi, kontrol listesine dayalı test, kapsam, karar kapsamı, karar tablosu testi, hata tahminleme, denklik paylarına ayırma, tecrübeye dayalı test tekniği, keşif testleri, durum geçişi testi, komut kapsamı, test tekniği, kullanım senaryosu testi, beyaz kutu test tekniği

Keywords
black-box test technique, boundary value analysis, checklist-based testing, coverage, decision coverage, decision table testing, error guessing, equivalence partitioning, experience-based test technique, exploratory testing, state transition testing, statement coverage, test technique, use case testing, whitebox test technique

Schlüsselbegriffe
Anweisungsüberdeckung, Anwendungsfallbasierter Test, Äquivalenzklassenbildung, Black-Box-Testverfahren, checklistenbasiertes Testen, Entscheidungstabellentest, Entscheidungsüberdeckung, erfahrungsbasierte Testverfahren, exploratives Testen, Grenzwertanalyse, intuitive Testfallermittlung, Testverfahren, Überdeckung, White-Box-Testverfahren, Zustandsübergangstest

4.1 Test Tekniklerinin Kategorileri

Test tekniğinin amacı, test koşullarını(test Conditions, Testbedingungen), test senaryolarını(test cases, Testfällen) ve test verilerini(test data, Testdaten) belirlemeye yardımcı olmaktır.

4.1.1 Test Tekniklerinin Seçimi

Hangi test tekniğinin kullanılacağı, aşağıdakiler de dâhil olmak üzere **bir dizi faktöre bağlıdır**:

- Birim veya sistem tipi
- Birim veya sistem karmaşıklığı
- Yasal standartlar
- Müşteri veya sözleşme gereksinimleri
- Risk seviyeleri
- Risk çeşitleri
- Test hedefleri
- Mevcut dokümantasyon
- Test uzmanının bilgi ve yetenekleri
- Kullanılabilir araçlar
- Süre ve bütçe
- Yazılım geliştirme yaşam döngüsü modeli

- Yazılımın beklenen kullanımı
- Test edilecek birim veya sistem üzerinde test tekniklerini kullanma konusunda geçmiş deneyim
- Birim veya sistemde beklenen hata çeşitleri

Bazı teknikler belirli durumlar ve test seviyeleri için daha uygundur; bazıları tüm test seviyelerine uygulanabilir.
Test senaryoları oluştururken test uzmanları en iyi sonuçları almak için test tekniklerinin genellikle bir kombinasyonunu kullanır.

Test analizi(test analysis), test tasarımı(test design) ve test uyarlama(test implementation) aktivitelerinde **test tekniklerinin kullanımı, gayri resmi (çok az veya sıfır dokümantasyon) ile resmi arasında değişebilir.**

Uygun resmilik seviyesi; test ve yazılım geliştirme süreçlerinin olgunluğu, zaman kısıtlamaları, emniyet veya yasal gereksinimler, proje ekibinin bilgi ve becerileri ve takip edilen yazılım geliştirme yaşam döngüsü modeli de dâhil olmak üzere test projesinin ve projenin bağlamına **bağlıdır.**

(Test analysis, Test design ve Test implementation aktivitelerinde test tekniklerinin kullanımı informelle formel arasında değişebilir. Formellik seviyesi; test ve yazılım geliştirme süreçlerinin olgunluğu, zaman kısıtlamaları, emniyet veya yasal gereksinimler, proje ekibinin bilgi ve becerileri ve takip edilen yazılım geliştirme yaşam döngüsü modeli de dâhil olmak üzere test projesinin ve projenin çeşidine bağlıdır)

4.1.2 Test Tekniklerinin Kategorileri ve Karakteristik Özellikleri

Bu ders programında test teknikleri kara kutu(black-box), beyaz kutu(white-box) veya tecrübeye dayalı(experience-based, erfahrungsbasierte) olarak sınıflandırılmaktadır.

Kara kutu test teknikleri (davranışsal veya davranışa dayalı teknikler(behavior-based techniques,spezifikationsbasierte Verfahren) olarak da bilinir) **uygun bir test esasının** (örneğin gereksinim dokümanları, spesifikasyonlar, kullanım senaryoları, kullanıcı hikâyeleri veya iş süreçleri(**business processes**).) **analizine dayanır.**

Bu teknikler **hem fonksiyonel hem de fonksiyonel olmayan testlere uygulanabilir.** Kara kutu test teknikleri, **test nesnesinin iç yapısını dikkate almadan test nesnesinin girdi ve çıktılarına odaklanır.**

(Davranışa dayalı tekniklerde(behavior-based techniques) diyebileceğimiz Black-box test bir Test base'in analizine dayanır.
Hem fonksiyonel hem de fonksiyonel olmayan testlere uygulanabilir. Kara kutu test teknikleri, test nesnesinin iç yapısını dikkate almadan test nesnesinin girdi ve çıktılarına odaklanır)

Beyaz kutu test teknikleri (yapısal veya yapıya dayalı teknikler(structure-based techniques,strukturbasierte Verfahren) olarak da bilinir), test nesnesinin mimarisinin(Architektur), ayrıntılı tasarımının, iç yapısının veya kodunun analizine dayanır. Kara kutu test tekniklerinden farklı olarak, beyaz kutu test teknikleri test nesnesinin içindeki yapı ve işlemlere odaklanır.

Tecrübeye dayalı test teknikleri(experience-based), testlerin tasarlanması, uyarlanması ve koşturulması için yazılımcıların, test uzmanlarının ve kullanıcıların tecrübelerini kullanır. Bu teknikler genellikle kara kutu ve beyaz kutu test teknikleriyle birleştirilir.

(Tecrübeye dayalı test teknikleri(experience-based), test uzmanlarının ve kullanıcıların tecrübelerini kullanır. Bu teknikler genellikle kara kutu ve beyaz kutu test teknikleriyle birleştirilir)

Kara kutu test tekniklerinin ortak özellikleri aşağıda verilmiştir:

- **Test koşulları, test senaryoları ve test verileri**; yazılım gereksinimlerini, spesifikasyonları, kullanım senaryolarını ve kullanıcı hikâyelerini içeren bir **test esasından elde edilir**.
- **Gereksinimlerden sapmaların** yanı sıra gereksinimler ve gereksinimlerin hayata geçirilmesi arasındaki boşlukları belirlemek için **test senaryoları kullanılabilir**.
- **Kapsam**, test esasındaki **test edilen öğelere** ve test esasına **uygulanan tekniğe göre ölçülür.**

(Test conditions, Test case ve Test datalar; software requirement, spesifikasyonlar, Testcase, user story leri içeren Test base'lerden elde edilir. Requirementlardan sapmaların tespiti için Test caseler kullanılabilir. Kapsam Test basedeki test edilen öğelere ve uygulanan tekniğe göre ölçülür)

Beyaz kutu test tekniklerinin ortak özellikleri aşağıda verilmiştir:

- **Test koşulları, test senaryoları ve test verileri;** kod, yazılım mimarisi, ayrıntılı tasarım veya yazılımın yapısına ilişkin diğer herhangi bir bilgiyi içeren **bir test esasından elde edilir.**
- **Kapsam**, seçilen bir yapı (örneğin, kod veya arayüzler) içerisinde **test edilen öğelere göre ölçülür.**
- Test senaryolarının beklenen çıktılarını belirlemek için genellikle ek bir bilgi kaynağı olarak spesifikasyonlar kullanılır.

(Test conditions, Testcase ve Testdataları; kod, software architecture, details design veya software structure'a ilişkin bilgi içeren bir Test base'den elde edilir.
Coverage(kapsam) kod veya interface içerisinde teste edilen öğeye göre ölçülür)

Tecrübeye dayalı test tekniklerinin(experience-based) ortak özellikleri aşağıda verilmiştir:

- Test koşulları, test senaryoları ve test verileri; test uzmanlarının, yazılımcıların, kullanıcıların ve diğer paydaşların bilgi ve tecrübelerini içeren bir test esasından elde edilir.

(Testconditions,Testcase ve Testdataları; test uzmanlarının, yazılımcıların, kullanıcıların ve diğer paydaşların bilgi ve tecrübelerini içeren bir test esasından elde edilir)

4.2 Kara Kutu Test Teknikleri

4.2.1 Denklik Paylarına Ayırma(Equivalence Partitioning, Äquivalenzklassenbildung)

Denklik paylarına ayırma (Equivalence Partitioning, Äquivalenzklassenbildung), verileri paylara (denklik sınıfları) ayırır; yazılım tarafından bir payın tüm üyelerinin benzer şekilde ele alınması

beklenir (bkz. Kaner 2013 ve Jorgensen 2014). Hem geçerli hem de geçersiz değerler için denklik payları vardır.

- Geçerli değerler(Valid values, Gültige Werte), birim veya sistem kapsamına giren, yazılım tarafından kabul edilmesi beklenen değerlerdir.
- Geçerli değerler içeren bir denklik payına "geçerli denklik payı(valid equivalence partition, gültige Äquivalenzklasse)" denir.
- Geçersiz değerler, birim veya sistem kapsamı dışında olan, yazılım tarafından reddedilmesi beklenen değerlerdir.
- Geçersiz değerler(Invalid values, Ungültige Werte) içeren bir denklik payına "geçersiz denklik payı(invalid equivalence partition, ungültige Äquivalenzklasse)" denir.
- Girdiler, çıktılar, dâhili değerler, zamana bağlı değerler (örneğin, bir olaydan önce veya sonra) dâhil test nesnesiyle ilgili herhangi bir veri öğesi için ve arayüz parametreleri için (ör. entegrasyon testleri sırasında test edilen entegre birimler) denklik payları belirlenebilir.
- Gerekirse bir pay alt-paylara ayrılabilir.
- Her değer yalnızca bir denklik payına ait olmalıdır, birden fazla payda yer almamalıdır.
- Test senaryolarında geçersiz denklik payları kullanıldığında arızaların maskelenmemesini sağlamak için bu paylar ayrı ayrı test edilmelidir, bir geçersiz denklik payı başka bir geçersiz denklik payıyla birleştirilmemelidir. Aynı anda birçok arıza oluştuğunda ancak yalnızca biri görünür olduğu için arızalar maskelenebilir ve diğer arızaların bulunmasına engel olur.

Bu teknikle %100 denklik payı kapsamı elde etmek için, test senaryoları, her paydan en az bir değer kullanarak, belirlenmiş tüm payları (geçersiz paylar dâhil) kapsamalıdır. Kapsam, test edilen denklik paylarının sayısının tüm denklik paylarının sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir. Denklik paylarına ayırma tüm test seviyelerinde uygulanabilir.

(Test case lerimizi seçerken aynı sonuçlara bizi ulaştıracağını düşündüğümüz parçaları tek parça olarak kabul edip o parçalardan bir tanesini deneyerek yapılan test tekniğidir. 3 ayrı cinsten üzüm kasasındaki üzümleri test etmek için 3 ayrı deneme yapmamız yeterlidir.

Ehliyet yaşı için 18 yaş sınırı vardır.18 yaştan küçük olanlar ve 18 yaş ve üstü olan 2 grup. Bunlardan birer örnek alarak bir tane valid, bir tanede invalid değerle 2 test yapmamız yeterlidir.
Sistem tarafından kabul edilen değerlere valid değerler denir.

Bir araba kiralama şirketi, 20 yaş altına ve 65 yaş üstüne araba kiralamama şartı koşmuş, 20 yaşın altı invalid bir değer, 20-65 yaş arası valid bir değer, 65 yaş üstü yine invalid bir değerdir. Burada 2 ayrı invalid bölümü var. Bunların her biri bireysel olarak test edilmesi lazım. Burada 3 tane equivalence partionımız var. %100 coverage kapsam sağlamak için en az 3 tane Test case yazmamız lazım. Sadece bir bölümün kontrolünü yaparsam %33 sağlamış olurum. %100 coverage sağlamak için her bölümden en az bir tanesinin test edilmesi lazım)

4.2.2 Sınır Değer Analizi(Boundary Value Analysis, Grenzwertanalyse)

Sınır değer analizi (Boundary Value Analysis(BVA), Grenzwertanalyse), denklik paylarına ayırma tekniğinin genişletilmiş halidir, ancak **yalnızca sayısal veya sıralı verilerden oluşan paylarda**

kullanılabilir. Bir payın minimum ve maksimum değerleri (veya ilk ve son değerleri) sınır değerleridir (Beizer 1990).

Örneğin, bir giriş alanının tek bir tam sayı değerini girdi olarak kabul ettiğini varsayalım, tam sayı olmayangirdilerin imkânsız olması için girdileri sınırlamakta bir tuş takımı kullanılıyor olsun. Geçerli aralık 1 ile 5 arasındadır, sınır değerleri de dâhildir.
Dolayısıyla, üç denklik payı vardır: geçersiz (alt); geçerli; geçersiz (üst). Geçerli denklik payı için sınır değerleri 1 ve 5'tir. Geçersiz (üst) denklik payı için sınır değerleri 6 ve 9'dur. Geçersiz (alt) pay için, yalnızca bir sınır değeri vardır, o da 0'dır, çünkü bu yalnızca bir üyesi olan bir paydır.
Yukarıdaki örnekte, sınır başına iki sınır değeri tanımlarız. Geçersiz (alt) ve geçerli arasındaki sınır, test değerleri olarak 0 ve 1'i verir. Geçerli ve geçersiz (üst) arasındaki sınır, test değerleri olarak 5 ve 6'yı verir.
Bu tekniğin bazı varyasyonları, sınır başına üç sınır değeri tanımlar: sınırdan önceki, sınırdaki ve hemen sınırdan sonraki değerler. Önceki örnekte, üç noktalı sınır değerleri kullanıldığında, alt sınır testi değerleri 0, 1 ve 2'dir ve üst sınır testi değerleri 4, 5 ve 6'dır (Jorgensen 2014).
Denklik paylarının sınırlarındaki davranışların hatalı olma olasılığı, payların içinden seçilip test edilecek verilere göre daha yüksektir. Yazılımın riskine göre sınır değerler üzerinde oynamalar yapmak; bunları kaydırmak, sınır başına seçilen sınır değer sayısını artırmak veya azaltmak mümkündür.
Sınır değer analizi ve testleri, test verilerini sınır değerin ait olduğu payın dışında farklı bir paydan seçerek yazılımın davranışlarını zorlamaktadır.
Sınır değer analizi tüm test seviyelerinde uygulanabilir. Bu teknik genellikle bir sayı aralığı (tarihler ve saatler dâhil) gerektiren gereksinimleri test etmek için kullanılır. Kapsam, test edilen sınır değerlerin sayısının tüm sınır değerlerin sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

(Bu teknik genellikle bir sayı aralığı (tarihler ve saatler dahil) gerektiren gereksinimleri test etme için kullanılır. Tüm test levellarında uygulanabilir.

Bir firmada her ayın 15'i ile 20'si arasında başvurular kabul ediliyor gibi bir requierement olsun, test edeceğimiz değer bu sınırlarda nasıl davanıyor bunu kontrol ederiz. Sınırların solunu ve sağını kontrol etmeye baundary analysis deniyor.

Iki yaklaşım var.two value boundary testing. three value bondary testing. Buna göre [1 - 5] arası değeri kontrol ettiğimizde
two value boundary : 2 değerli sınır testi 0 ve 1, 5 ve 6
three value boundary testing. 3 değerli sınır testi denirse. 0,1,2 ve 4,5,6

Boundary value analysis normalde iki değerli yapılır. 3 değerli value analysis denilirse 3'lü olarak yapıyoruz)

4.2.3. Karar Tablosu Testleri (Decision Table Testing, Entscheidungstabellentests)

Karar Tablosu Testlerinin(Decision Table Testing, Entscheidungstabellentests) de yer aldığı **kombinasyonlu test teknikleri farklı test koşulları kombinasyonlarının nasıl farklı sonuçlar verdiğini test etmek için kullanılır.**

Karar tabloları, bir yazılımın ele alması beklenen **karmaşık iş kurallarını kaydetmenin iyi bir yoludur**. Karar tabloları oluştururken, test uzmanı yazılımın koşullarını(conditions, Bedingungen) (genellikle girdiler) ve sonuçta ortaya çıkan aksiyonları(actions(Result), Aktionen) (genellikle çıktılar) **tanımlar**. Bunlar tablonun satırlarını oluşturur, genellikle koşullar üstte ve aksiyonlar alttadır. Her sütun bir karar kuralına karşılık gelir; bu kural, kendisiyle ilişkili aksiyonların gerçekleştirilmesini sağlayan özgün bir koşul kombinasyonunu içerir. Koşulların ve aksiyonların değerleri genellikle mantıksal değerler (boolean - doğru veya yanlış) veya ayrık değerler (örneğin kırmızı, yeşil, mavi) olarak gösterilir, ancak sayılar veya sayı aralıkları da olabilir. Bu farklı türlerdeki koşullar ve aksiyonlar aynı tabloda bir arada bulunabilir.
Karar tablolarındaki yaygın gösterim aşağıdaki gibidir:
Koşullar için:
Y, koşulun doğru olduğu anlamına gelir (ayrıca T veya 1 olarak da gösterilebilir)
N, koşulun yanlış olduğu anlamına gelir (ayrıca F veya 0 olarak da gösterilebilir)
— koşulun değerinin önemli olmadığı anlamına gelir (N/A olarak da gösterilebilir) Aksiyonlar için:
X, eylemin gerçekleşmesi gerektiği anlamına gelir (ayrıca Y veya T veya 1 olarak da gösterilebilir)
Boşluk, eylemin gerçekleşmemesi gerektiği anlamına gelir (ayrıca - veya N veya F veya 0 olarak da gösterilebilir)
Tam bir karar tablosunda her koşul kombinasyonunu kapsayacak kadar sütun vardır. **İmkânsız koşul kombinasyonlarını içeren sütunları, olası ancak elverişsiz koşul kombinasyonlarını içeren sütunları ve çıktıyı etkilemeyen koşulların kombinasyonlarını test eden sütunları silinerek tablo küçültülebilir.** Karar tablolarının nasıl küçültüleceği hakkında daha fazla bilgi için, bkz. ISTQB-ATA İleri Seviye Test Analisti Ders Programı.

Karar tablosu testleri için genel asgari kapsam standardı, **tablodaki her karar kuralı için en az bir test senaryosun yazılmasıdır.** Bu, genellikle tüm koşul kombinasyonlarının kapsanmasını içerir. Kapsam, test edilmiş karar kurallarının sayısının, karar kurallarının toplam sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

Karar tablosu testlerinin güçlü yanı, hiçbir koşulu göz ardı etmeden tüm önemli koşul kombinasyonlarını ele almayı sağlamasıdır. Aynı zamanda bu teknik gereksinimlerdeki boşlukların bulunmasında da yardımcı olur. Her test seviyesinde, yazılım davranışının koşulların bir kombinasyonuna bağlı olduğu tüm durumlarda uygulanabilir.

(Daha komplex yapıları çözerken bu tekniği kullanıyoruz.
Örnek : Bir araba kiralama şirketi 20 ile 65 yaş arasına arabalarını kiralıyor. Şirket müracaat eden şahıs daha öncesinden herhangi bir cezası yoksa %15 indirim, aracı iş maksatlı kiralıyorsa artı %10 daha indirim yapıyor. Bu şekilde komplex tanımlamalar olduğunda Decision Table Testing yöntemini kullanıyoruz.

Sistemin gözardı edemeyeceği önce şartlar, condition olarak tanımlanıyor. Belirtilen koşullara göre bir result veya action ortaya çıkıyor. Bu şekilde yazılan tablolar Decision rule diye adlandırılıyor.
Şartlar condition lar tanımlanmış
Test case : true(Y, T, 1) veya false(N, F, 0)
Bunlara bağlı olarak bir result ortaya çıkıyor.
Şart true veya false olmayan bir değer olduğu zaman - veya N/A olarak yazılır.

Örnek:
1- işe alınan 1 yıldırı aşkındır işte çalışıyorsa
2- bu yıl 500 kalem mal satarsa bonus kazansın
3- Bu şekilde bir hedef var mı ve
4- Bu hedefe ulaşılmış mı, bu şartlardan biri sağlanıyorsa bonus ödeniyor, sağlanmıyorsa ödenmiyor
Hesap aktif değil
ödeme işlemi yapma
hesap sahibini bilgilendirme
satıcıyı bilgilendir gibi...

Kaç tane condition var sayıyoruz, eğer 3 tane condition varsa 2 üzeri 3 tane test case yazmamız gerekiyor. yani 8. 8 Test case'i bir tablo halinde yukarıdan aşağıya;
4 tane yes 4 tane no
2 defa no 2 defa yes
bir yes bir no
Olacak şekilde yazıp, her bir Test case sütunun altına bu combinasyonlara göre sonuçların Y veya N şeklinde yazıyoruz.

4 condition'ın çalışabilmesi için 4 şartta yes olması lazım.
4 condition olsaydı 16 test case yazacaktık.

Sonuçlar aynı ve conditionlardan sadece bir tanesi farklı ise farklı olan condition'ın sonuca etkisi yok(N/A) demektir. Bu gibi durumlarda benzer Test caselerden sadece bir tanesine çalışmamız yeterli diğerlerini eleyip Test case sayısını azaltabiliriz.

GEREKSİZ TEST CASE YAZMAMAK İÇİN:
Actions'a(result) bakıyorum, birbirinin aynı olanları buluyorum, sonra şartları(condition) inceliyorum, burada birbirinden farklı sadece bir Test case uyuşmazlığı varsa Test case sayısını düşürebiliyoruz.

1- Gerçek hayatta karşılığı olmayan mantıksız Test case'leri yazmamıza gerek yok.
2- Aynı sonucu veren benzer Test case'lerin hepsini yazmamıza gerek yok)

4.2.4 Durum Geçişi Testleri(State Transition Testing, Zustandsübergangstest)

Birimler veya sistemler, mevcut veya geçmiş durumuna (örneğin, sistemin başlatılmasından bu yana gerçekleşen olaylar) bağlı olarak gerçekleşen bir olaya farklı şekilde yanıt verebilir. Bir sistemin geçmiş

davranışı, durumlar kavramı kullanılarak özetlenebilir. Durum geçişi diyagramı(state transition diagram, Zustandsübergangsdiagramm) **olası yazılım durumlarının yanı sıra, yazılımın ilgili durumlara nasıl girdiğini, çıktığını ve aralarındaki geçişleri gösterir.** Geçiş, olay tarafından başlatılır (örneğin, bir değerin bir alana kullanıcı tarafından girilmesi).

Olay, bir aksiyonla sonuçlanır. Aynı olay aynı durumdan iki veya daha fazla farklı geçişle sonuçlanabilirse, bu olay bir koruma koşulu olarak nitelendirilebilir. Durum değişikliği, yazılımın bir işlem gerçekleştirmesine neden olabilir (örneğin bir hesaplama veya hata mesajı çıktısı oluşturma).

Durum geçişi tablosu, durumlar arasındaki tüm geçerli geçişleri ve potansiyel geçersiz geçişleri, ayrıca olayları, koruma koşullarını ve geçerli geçişler için aksiyonları gösterir. **Durum geçiş diyagramları normalde yalnızca geçerli geçişleri gösterir ve geçersiz geçişleri ele almaz.**
Tipik bir durum dizisini kapsamak, tüm durumları denemek, her geçişi denemek, belirli geçiş dizilerini denemek veya geçersiz geçişleri test etmek için testler tasarlanabilir.

Durum Geçişi Testleri(State Transition Testing , Zustandsübergangstest), **menü bazlı uygulamalar için kullanılabilir ve gömülü yazılım endüstrisinde yaygın bir şekilde kullanılmaktadır.** Bu teknik ayrıca **belirli durumlara sahip bir iş senaryosunu modellemek veya ekran geçişlerini test etmek için de uygundur.** Durum, soyut bir kavramdır - birkaç kod satırını veya bütün bir iş sürecini temsil edebilir.
Kapsam, genel olarak test edilen durum veya geçişlerin sayısının, test nesnesindeki durumların veya geçişlerin toplam sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir. Durum geçişi testlerinin kapsam kriterleri hakkında daha fazla bilgi için, bkz. ISTQB-ATA İleri Seviye Test Analisti Ders Programı.

(Bir durumdan başka bir duruma geçişe transition deniyor, bunu test etmeye Durum Geçişi Testleri(State Transition Testing , Zustandsübergangstest) denir.
Login sayfası doğru bilgileri girince giriş sayfasını bize taşımalı yoksa error message vermeli, bu iki davranışa durum deniyor. Bu testle asıl amçlanan her bir durumun sergilediği davranışların kontrolünü yapmak.
Menü bazlı uygulamalar için kullanılabilir ve gömülü yazılım endüstrisinde yaygın bir şekilde kullanılmaktadır. Bu teknik ayrıca belirli durumlara sahip bir iş senaryosunu modellemek veya ekran geçişlerini test etmek için de uygundur. Durum, soyut bir kavramdır - birkaç kod satırını veya bütün bir iş sürecini temsil edebilir)

4.2.5 Kullanım Senaryosu Testleri (Use Case Testing, Anwendungsfallbasierter Test)

Testler, **farklı profildeki kullanıcıların yazılımla etkileşimleri** modellemek için faydalanılan kullanım senaryolarından elde edilebilir; **bu testler, kullanıcı gereksinimlerini ele alır**. Kullanım senaryoları, aktörler (kullanıcılar, harici donanım, diğer birim veya sistemler) ve sistemlerle (kullanım senaryosunun uygulandığı birim, sistem veya yazılımla) ilişkilidir.

Her kullanım senaryosu, bir sistemin bir veya daha fazla aktör ile işbirliği içinde gerçekleştirebileceği bazı davranışları belirtir (UML 2.5.1 2017). Bir kullanım senaryosu, etkileşimler ve aktivitelerin yanı sıra uygun durumlarda önkoşullar, artkoşullar ve doğal bir anlatımla tanımlanabilir. **Aktörler ve sistem arasındaki etkileşimler sistemin durumunda değişikliklere neden olabilir**. Etkileşimler; iş akışları, faaliyet şemaları veya iş süreç modelleri ile grafiksel olarak gösterilebilir.
Bir kullanım senaryosu, istisnai davranış ve hata ele alma dâhil olmak üzere, temel yazılım davranışının olası varyasyonlarını, alternatiflerini içerebilir. Testler, tanımlanmış davranışları (temel, alternatif, istisnai ve hata ele alma) denemek için tasarlanır. Kapsam, test edilen kullanım senaryosu davranışlarının toplam kullanım senaryosu davranışı sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.
Kullanım senaryosu testlerinin kapsam kriterleri hakkında daha fazla bilgi için, bkz. ISTQB-ATA İleri Seviye Test Analisti Ders Programı.

(Kullanım Senaryosu Testleri (Use Case Testing, Anwendungsfallbasierter Test), aktif çalışan bir sistem üzerinde test edildiği için sadece System test ve Acceptence testler için bu testi kullanabiliyoruz. Kullanıcı durumlarından testler oluşturulur.
User var birde sistem var, user davranışına göre sistem bir davranış sergiler. Mesela atm ye gittik kartı girdik, sistemin çalışıyor olduğunu bilmemiz için ekranda yazıyı okumamız lazım bunun için önce sistemin hazır olması lazım ki, sistemi test edebilelim. Bütün işlemler bittikten sonra kartı teslim aldıktan sonra size good bye demesi lazım, bu sistemin bir davranışıdır.
use case çalışılırken sistemle iletişime geçen basic davranışlar gözününde bulundurulur ama bazen exeptional behavior(beklenmeyen, istisna bir davranış) sergilemesi lazım, sisteme kart yerine ehliyet soktunuz, exeptional behavior gösterdi bu bize sistemin çalıştığını gösteren farklı bir test yöntemidir)

4.3 Beyaz Kutu Test Teknikleri

Beyaz kutu testleri, test nesnesinin iç yapısına (internal structure, internen Struktur) dayanır. Beyaz kutu test teknikleri tüm test seviyelerinde kullanılabilir, ancak bu bölümde açıklanan koda bağlı iki teknik en yaygın şekilde birim test seviyesinde kullanılır. Daha eksiksiz bir kapsama elde etmek için bazı hayati açıdan kritik veya çok fazla entegrasyonun olduğu ortamlarda kullanılan daha ileri teknikler de vardır, ancak bunlar burada ele alınmayacaktır. Bu tür teknikler hakkında daha fazla bilgi için ISTQB İleri Seviye Teknik Test Analisti ders programına bakınız.

(White box testleri , test objesinin iç yapısına(internal structure) dayanır. Tüm test seviyelerinde kullanılabilir. Koda bağlı iki teknik(Statement ve Decision testing) en yaygın şekilde component test seviyesinde kullanılır)

4.3.1 Komut Testi ve Kapsamı(Statement Testing and Coverage, Anweisungstests und -überdeckung)

Komut testi (Statement Testing, Anweisungstests) kod içinde yer alan yürütülebilir komutların üzerinden geçilip bu komutların çalıştırılmasıdır(executed). Kapsam, testler tarafından çalıştırılan komutların sayısının test nesnesindeki çalıştırılabilir komutların toplam sayısına bölünmesi ile ölçülür ve normalde yüzde olarak ifade edilir.

(white box tüm test levellarında kullanılabilir, dolayısıyla sadece developerlar değil testerlarda code üzerinde test yaparak white box test yapabilir. Statement testing kod içinde yer alan Execute edilebilecek statementların(komutların) üzerinden geçilip çalıştırılmasıdır)

4.3.2 Karar Testi ve Kapsamı(Decision Testing and Coverage, Entscheidungstest und -überdeckung)

Karar testi (Decision Testing, Entscheidungstest), koddaki kararların(statement) üzerinden geçilip bu kararların çalıştırılması ve karar çıktılarına dayanarak kodun test edilmesidir. Bunun için test senaryoları karar noktasındaki kontrol akışlarını takip eder (örneğin, bir IF komutu için, biri doğru çıktı ve biri yanlış çıktı; bir CASE komutu için tüm olası çıktıların test senaryoları gerekli olacaktır).
Kapsam, testler tarafından çalıştırılan karar çıktılarının sayısının test nesnesindeki karar çıktılarının toplam sayısına bölünmesi ile ölçülür ve normalde yüzde olarak ifade edilir.

(Bir if Statement düşünelim, şart sağlanıyorsa true olacak ona göre işlem gerçekleştirecek değilse false olup ona göre farklı bir işlem gerçekleştirecek. Burada bir karar verecek buna Decision diyoruz, bu Decision'ın test edilmesine de Decision test diyoruz.

If close'ın her iki tarafını en az bir kere kontrol ederseniz, %100 statement coverage , sadece bir tarafı kontrol etmiş olsak %50 coverage sağlamış oluruz.
Aşağıdakilerden hangisi white box testtir?
Decision table testing: Black box testi

Decision testing: white box testi
Statement Transition testing: Black box testi
Statement Testing : White box testi)

4.3.3. Komut(Statement) ve Karar(Decision) Testlerinin Önemi

%100 komut kapsama yüzdesi sağlandığında koddaki tüm çalıştırılabilir komutların en az bir kez test edilmesi sağlanır,ancak tüm kararların tamamen test edilmesi sağlanmaz. Bu ders programında bahsedilen iki beyaz kutu tekniği arasında komut testleri, karar testlerine kıyasla daha az kapsama sağlayabilir.
%100 karar kapsamı sağlandığında, tüm karar çıktıları çalıştırılır; buna, doğru çıktının ve yanlış çıktının test edilmesi de dâhildir. Komut kapsama yüzdesi, kodda, diğer testler tarafından denenmemiş hataların bulunmasına yardımcı olur. Karar kapsamı, ise diğer testlerin ele almadığı hem doğru hem de yanlış çıktıların ele alınmasını sağlar.
%100 karar kapsamına ulaşılması %100 komut kapsamına ulaşılmasını garanti eder (ancak bunun tersi geçerli değildir).

(Statement testing decision testten daha az bir coverage sağlar. %100 decision coverage sağladığınızda %100 statement coverage sağlamış sayılırsınız ama tam tersi geçerli değildir
white box non functinal değildir diye ifade etmek doğru değil.

Decision Testing için Condition Coverage ismide kullanılabiliyor, sinonim ifadeler.
kare ile ifade edilme statement
baklava dilimi : Decision
oklar: edges, path, yollar

3 statement var.
eğer %100 statement coverage sağlamak istiyorsam bu 3 statementtada çalışmam gerekiyor. bunlarla ilgili test case yazmam gerekiyor. şema üzerinde çalışırken önce kaç test case yazacağımızı belirliyoruz. Bazen bir tane test case ile 3 statement coverage'ı test ederek %100 coverage sağlanabilir.

%100 path(güzergah, yol) diye bir şey sorarsa bütün yolları test edilmesi gerektiğini anlıyoruz.

Decision coverage dediğimizde aklımıza if gelmeli.

%100 Decision coverage için kararların hem yes hem no kısmını çalışmamız gerekiyor.

branche coverage deyince decision coverage olarak anlıyoruz

her zaman Decision test sayısı >= Statement test sayısı

full path coverage: tüm yollardan geçilmeli
%100 statement coverage
%100 Decision coverage: hem false, hem truedan geçilmeli

sorularda bu üçü sorulabilir.

end if deyince hem true hem false sonlanıyor. tam bir çizgi çek)

4.4 Tecrübeye Dayalı Test Teknikleri(Experience-based Test Techniques, Erfahrungsbasierte Testverfahren)

Tecrübeye dayalı test teknikleri (Experience-based Test Techniques, Erfahrungsbasierte Testverfahren) uygulanırken, test senaryoları, test uzmanının beceri ve sezgilerinden ve benzer uygulama ve teknolojilerdeki tecrübelerinden elde edilir. **Bu teknikler, diğer daha sistematik tekniklerle kolayca bulunamayan hataları bulmada yardımcı olabilir. Test uzmanının yaklaşımına ve tecrübesine bağlı olarak**, bu teknikler çok çeşitli kapsam ve etkinlik derecelerine ulaşabilir. Kapsamın değerlendirilmesi zor olabilir ve bu tekniklerle ölçülemeyebilir.
Yaygın olarak kullanılan tecrübeye dayalı teknikler aşağıdaki bölümlerde ele alınmaktadır.

(Experience-based Testte test senaryoları, test uzmanının beceri ve sezgilerinden ve benzer uygulama ve teknolojilerdeki tecrübelerinden elde edilir. Test uzmanının yaklaşımına ve tecrübesine bağlı olarak diğer daha sistematik tekniklerle bulunamayan veya çok efor sarf edilerek bulunabilecek hatalar bu teknik kullanılarak daha kolay bir şekilde bulunabilir)

4.4.1 Hata Tahminleme(Error Guessing, Intuitive Testfallermittlung)

Hata tahminleme (Error Guessing, Intuitive Testfallermittlung), test uzmanının bilgilerine dayalı olarak yanlışların(error), hataların(defect) ve arızaların(failures) oluşmasını sağlamak için kullanılan bir tekniktir; bu bilgiler aşağıdaki gibidir:

- Uygulamanın geçmişte nasıl çalıştığı
- Yazılımcıların yapmaya eğilimli oldukları hata çeşitleri
- Diğer uygulamalarda oluşan arızalar

Hata tahminleme(Error Guessing) tekniğine metodolojik bir yaklaşım, olası yanlışların, hataların ve arızaların bir listesini oluşturmak ve bu arızaları ve bunlara neden olan hataları ortaya çıkaracak testler tasarlamaktır. Bu yanlış, hata, arıza listeleri tecrübeye, hata ve arıza verilerine dayanarak veya yazılımların neden başarısız olduğu hakkındaki genel bilgilerden yola çıkarak oluşturulabilir.

(Hata tahminleme (Error Guessing, Intuitive Testfallermittlung), test uzmanının bilgilerine dayalı olarak yanlışların(error), hataların(defect) ve arızaların(failures) oluşmasını sağlamak için kullanılan bir tekniktir.
informal, tecrubeye dayalı bir test tekniğidir. Tester, developerlar hangi çeşit errorlara meyilli tecrubeye dayalı olarak bu kısımlara yoğunlaşır, bu hatalara sebep olan defectleri yakalamak için testlerini dizayn eder, hatanın nerede olduğunu tahmin eder, spesifik olarak gider orayı kontrol eder, hatayı bulmak için hangi test tekniğini kullanmak isterse onu seçer ve test eder)

4.4.2 Keşif Testi(Exploratory Testing, Exploratives Testen)

Keşif testlerinde (Exploratory Testing, Exploratives Testen), test koşumu sırasında **gayri resmi** (önceden tanımlanmamış) testler tasarlanır, koşturulur, kaydedilir ve dinamik olarak değerlendirilir. **Test**

sonuçları, birim veya sistem hakkında daha fazla bilgi edinmek ve daha fazla test gerektirebilecek alanlar için testler oluşturmak için kullanılır.

Keşif testi sürecini daha sistematik yapmak için oturuma dayalı testlerden faydalanılır. Oturuma dayalı testlerde, keşif testleri **belirli bir zaman dilimi içinde gerçekleştirilir** ve test uzmanı, testi yönlendirmek için test hedeflerini içeren bir test tüzüğünü kullanır. Test uzmanı, izlenen adımları ve yapılan keşifleri dokümante etmek için notlar alabilir.

Keşif testleri, gereksinimler az veya yetersiz olduğunda veya testler üzerinde önemli bir zaman baskısı olduğunda çok işe yarar. Keşif testleri, diğer daha resmi test tekniklerini tamamlamak için de kullanılır.
Keşif testleri, tepkisel test stratejileri(reactive test strategies) ile güçlü bir şekilde ilişkilidir (bkz. Bölüm 5.2.2). Keşif testleri, diğer kara kutu, beyaz kutu ve tecrübeye dayalı tekniklerle birlikte kullanılabilir.

(Exploratory Testing informal olarak yapılır, sonuçları component veya sistem hakkında daha fazla test gerektirebilecek alanları tespit edip testler oluşturmak için kullanılır. Gereksinimler az veya yetersiz olduğunda veya testler üzerinde önemli bir zaman baskısı olduğunda çok işe yarar. Tepkisel test stratejileri(reactive test strategies) ile güçlü bir şekilde ilişkilidir)

4.4.3 Kontrol Listesine Dayalı Testler(Checklist-based Testing, Checklistenbasiertes Testen)

Kontrol listesine dayalı testlerde (Checklist-based Testing, Checklistenbasiertes Testen), test uzmanları **bir kontrol listesi**nde bulunan test koşullarını kapsayacak şekilde **testler tasarlar, uygular ve koşturur.** Analizinin bir parçası olarak, test uzmanları yeni bir kontrol listesi oluşturur veya mevcut bir kontrol listesini genişletir, ancak test uzmanları mevcut bir kontrol listesini değişiklik yapmadan da kullanabilir. Bu kontrol listeleri **tecrübe, kullanıcı için neyin önemli olduğu veya yazılımın niçin ve nasıl başarısız olabileceği bilgisine dayanarak oluşturulabilir.**

Fonksiyonel ve fonksiyonel olmayan testleri desteklemek için kontrol listeleri oluşturulabilir. Ayrıntılı test senaryoları olmadığında, kontrol listesine dayalı testler rehberlik ve belirli bir yere kadar tutarlılık sağlayabilir. Bunlar üst seviye listeler olduğu için, gerçek testlerde bazı değişikliklerin ortaya çıkması muhtemeldir; bu da potansiyel olarak daha büyük kapsama ama daha düşük tekrarlanabilirliğe yol açar.

(Mevcut veya yeni oluşturulan bir checklistte yazılanların karşılanıp karşılanmadığı testler tasarlanak kontrol edilir. Checklistler tecrübe, kullanıcı için neyin önemli olduğu veya yazılımın niçin ve nasıl başarısız olabileceği bilgisine dayanarak oluşturulur.
Fonksiyonel ve fonksiyonel olmayan testleri desteklemek için kontrol listeleri oluşturulabilir. Ayrıntılı test senaryoları olmadığında, kontrol listesine dayalı testler rehberlik ve belirli bir yere kadar tutarlılık sağlayabilir)

Chapter 5

5. Test Yönetimi(Test Management)

Anahtar kelimeler
yapılandırma yönetimi, hata yönetimi, giriş kriterleri, çıkış kriterleri, ürün riski, proje riski, risk, risk seviyesi, risk-bazlı test, test yaklaşımı, test kontrolü, test tahminlemesi, test yöneticisi, test gözetimi, test planı, test planlama, test ilerleme raporu, test stratejisi, test özet raporu, test uzmanı

Keywords
configuration management, defect management, defect report, entry criteria, exit criteria, product risk, project risk, risk, risk level, risk-based testing, test approach, test control, test estimation, test manager, test monitoring, test plan, test planning, test progress report, test strategy, test summary report, tester

Schlüsselbegriffe
Eingangskriterien, Endekriterien, Fehlerbericht, Fehlermanagement, Konfigurationsmanagement, Produktrisiko, Projektrisiko, Risiko, risikobasiertes Testen, Risikostufe, Testabschlussbericht, Tester, Testfortschrittsbericht, Testkonzept, Testmanager, Testplanung, Testschätzung, Teststeuerung, Teststrategie, Testüberwachung, Testvorgehensweise

5.1 Test Organizasyonu(Testorganisation)

5.1.1Testte Bağımsızlık (Independent Testing, Unabhängiges Testen)

Test görevleri, belirli bir test rolündeki kişiler veya başka bir roldeki kişiler tarafından (ör. müşteriler) yerine getirilebilir. **Belli bir ölçüde bağımsızlık**, test uzmanının **hata bulmasını daha etkili kılar** (bkz. Bölüm 1.5). Ancak **bağımsızlık, aşina olmanın yerine geçemez** ve **yazılımcılar kendi kodlarındaki birçok hatayı verimli bir şekilde bulabilir.**

(Bağımsızlık belli ölçüde hata bulmayı etkili kılabilir ama aşina olmanın yerine geçemez. Yazılımcılar kendi kodlarındaki hatayı verimli bir şekilde bulabilir)

Testlerdeki bağımsızlık dereceleri (düşük bağımsızlık seviyesinden yüksek seviyeye) aşağıda sıralanmıştır:

- Bağımsız test uzmanı olmadan; yazılımcıların kendi kodlarını test etmesi
- Yazılımcıların diğer yazılımcıların kodlarını test etmesi veya yazılım geliştirme ekipleri veya proje ekibi içinde yer alan test uzmanlarının yazılımcıların kodlarını test etmesi
- Kurum içinde, proje yönetimi veya üst düzey yöneticilere bağlı çalışan bağımsız test ekibi veya grubunun olması
- Kurumdan veya kullanıcılardan seçilen test uzmanlarının testleri yapması veya kullanılabilirlik, güvenlik, performans, yasal/uyumluluk veya taşınabilirlik gibi belirli test çeşitlerinde uzmanlıklara sahip bağımsız test uzmanlarının testleri yapması

- Kurumun dışından gelen bağımsız test uzmanlarının ekip içine dahil edilerek veya ekip dışında yer alarak testleri yapması

(Düşük seviyden yüksek seviyeye doğru Testlerdeki bağımsızlık dereceleri; yazılımcıların kendi kodlarını, devlp diğer dvlpr'ın kodlarını veya aynı ekipteki testerların kodları test etmesi, üst düzey yöneticilere bağlı çalışan bağımsız test ekibi tarafından testlerin yapılması, kurumdan veya kullanıcılardan seçilen test uzmanlarının testleri yapması, kurum dışından gelen test uzmanları tarafından testlerin yapılması)

Çoğu proje türü için, birden fazla test seviyesi olması ve bu seviyelerin bir kısmının bağımsız test uzmanları tarafından ele alınması genellikle en iyisidir. Yazılımcılar, kendi çalışmalarının kalitesi üzerinde kontrol sahibi olmak için özellikle alt seviyelerde testlere katılmalıdır.

Testlerin bağımsızlığının uygulanma şekli, yazılım geliştirme yaşam döngüsü modeline göre değişir. Örneğin, **çevik yazılım geliştirmede, test uzmanları yazılım geliştirme ekibinin bir parçası olabilir.** Çevik çerçeveler kullanan bazı kurumlarda, bu test uzmanları daha büyük bir bağımsız test ekibinin parçası olarak kabul edilebilir. Ek olarak, bu tür kurumlarda, ürün sahipleri, her döngü(iteration) sonunda kullanıcı hikâyelerinin(**user story) sağlamasını yapmak için kabul testleri(acceptance testing) gerçekleştirebilir.** ***(Testlerinin bağımsızlığının uygulanma şekli SDLC modeline göre değişir)***

Test bağımsızlığının potansiyel faydaları aşağıda sıralanmıştır:

- Farklı **geçmiş deneyimleri**, **teknik bakış açıları** ve **eğilimleri** nedeniyle **bağımsız test uzmanlarının, yazılımcılara kıyasla farklı çeşitteki arızaları bulma olasılıkları yüksektir**.
- Bağımsız bir test uzmanı, sistemin spesifikasyonunun belirlenmesi ve uygulanması sırasında paydaşların yaptığı varsayımları doğrulayabilir, test edebilir veya çürütebilir.

Test bağımsızlığının potansiyel sakıncaları aşağıda sıralanmıştır:

- Yazılım geliştirme ekibinden ayrı durma; iş birliği eksikliği, yazılım geliştirme ekibine geri bildirim sağlamada gecikmeler veya yazılım geliştirme ekibiyle düşmanca ilişkilere yol açabilir
- Yazılımcılar kalite konusunda sorumluluk bilincini kaybedebilir
- Bağımsız test uzmanları bir dar boğaz olarak görülebilir veya sürümdeki gecikmelerden sorumlu tutulabilir.
- Bağımsız test uzmanları (örneğin test nesnesi hakkında) bazı önemli bilgilere sahip olmayabilir.

Birçok kurum, test bağımsızlığının dezavantajlarını minimuma indirirken, yararlarından faydalanmayı bilmişlerdir.

(Test bağımsızlığının faydaları; Geçmiş deneyim, teknik bakış açıları ve eğilimleri nedeniyle testerlar developerlara kıyasla farklı çeşitteki arızaları bulma olasıları yüksektir. Sistemin kendine has özelliklerde Stakeholder'ların yaptığı varsayımları doğrulayabilir, test edebilir veya çürütebilir.
Potansiyel sakıncaları; ekip ruhunun oluşamaması, developerlar kalite konusunda sorumluluk bilincini kaybedebilir, test uzmanları işlerin ilerlemesine engel, gecikmelerden sorumlu tutulabilir)

5.1.2 Test Yöneticisi ve Test Uzmanının Görevleri

Bu ders programında iki test rolünden bahsedilmektedir; test yöneticisi ve test uzmanı. Bu iki role sahip kişiler tarafından gerçekleştirilen faaliyetler ve görevler, **projenin ve ürünün bağlamına, o rolü üstlenen kişilerin becerilerine ve kuruma göre değişir.**

Test yöneticisi, genel olarak test **sürecinden** ve test **faaliyetlerine başarılı bir şekilde liderlik** edilmesinden sorumludur. Test yöneticisi rolü; **profesyonel bir test yöneticisi**, **proje yöneticisi, yazılım geliştirme yöneticisi veya kalite güvence yöneticisi tarafından gerçekleştirilebilir.** Daha büyük projelerde veya kurumlarda, bir test yöneticisine, test koçuna veya test koordinatörüne bağlı olarak birçok test ekibi çalışabilir ve her ekip, bir test lideri veya deneyimli test uzmanı tarafından yönetilir.

Tipik test yöneticisinin görevleri aşağıdakileri içerebilir:

- Kurum için **test politikası ve test stratejisi geliştirmek veya bunları gözden geçirmek**
- Proje ve test bağlamını göz önünde bulundurarak ve test hedeflerini ve risklerini anlayarak **test faaliyetlerini planlamak**. Bu, test yaklaşımlarının seçilmesini, test zamanını, çalışmasını ve maliyetini tahminlemeyi, kaynakları sağlamayı, test seviyelerini ve test döngülerini tanımlamayı ve **hata yönetimini planlamayı içerebilir**.
- **Test planını/planlarını oluşturmak** ve güncellemek
- Proje yöneticileri, ürün sahipleri ve diğer paydaşlarla birlikte **test planını/planlarını koordine etmek**
- Test bakış açılarını diğer proje faaliyetleriyle paylaşmak, örneğin **entegrasyon planlama**
- **Testlerin analizini, tasarımını, uyarlanmasını ve koşturulmasını başlatmak, testin ilerlemesini ve sonuçlarını izlemek ve çıkış kriterlerinin durumunu (tamamlandı tanımını) kontrol etmek**
- Elde edilen bilgilere dayanarak **test ilerleme raporları ve test özet raporları hazırlamak ve göndermek**
- Test sonuçlarına ve ilerlemesine dayanarak **planlamayı revize etmek** (bazen test ilerleme raporlarında ve/veya projede önceden tamamlanmış olan diğer testler için test özet raporlarında dokümante edilir) ve **test kontrolü için gerekli önlemleri almak**
- Hata yönetimi sistemi ve test yazılımı için gerekli yapılandırma yönetiminin kurulmasını desteklemek
- Test **ilerlemesini ölçmek**, testlerin ve **ürünün kalitesini değerlendirmek** için uygun **metrikler oluşturmak**
- Araç seçimi için **bütçe önerme** (ve muhtemelen satın alma ve/veya destek), pilot projeler için zaman ve çalışma ayırma ve aracın/araçların kullanımında sürekli destek sağlama da dâhil olmak üzere **test sürecini destekleyecek araçların seçimini ve uygulanmasını desteklemek**
- **Test ortamının/ortamlarının uyarlanması hakkında karar vermek**
- Kurum içinde **test uzmanlarını**, test ekibini ve test uzmanlığı mesleğini **tanıtmak ve savunmak**
- Test uzmanlarının **becerilerini ve kariyerlerini geliştirmek** (örneğin, eğitim planları, performans değerlendirmeleri, koçluk vb ile)

Test yöneticisi rolünün yerine getirilme şekli, yazılım geliştirme yaşam döngüsü modeline göre değişir. Örneğin, çevik geliştirmede, yukarıda belirtilen görevlerin bazıları çevik ekip tarafından gerçekleştirilir; özellikle ekip içinde yapılan günlük testlerle ilgili görevler, genellikle ekip içinde çalışan bir test uzmanı tarafından yerine getirilir. Birden fazla ekibi veya tüm kurumu kapsayan veya personel yönetimi ile ilgili

olan bazı görevler, yazılım geliştirme ekibi dışından test yöneticileri tarafından yapılabilir; bunlara bazen test koçları denir.

(Test manager'ın görevleri; Proje yöneticileri, ürün sahipleri ve diğer paydaşlarla birlikte test planını/planlarını koordine etmek.Test ilerlemesini ölçmek, testlerin ve ürünün kalitesini değerlendirmek için uygun metrikler oluşturmak.
Test yöneticisi rolünün yerine getirilme şekli, yazılım geliştirme yaşam döngüsü modeline göre değişir)

Test sürecinin yönetimi hakkında daha fazla bilgi için Black 2009'a bakınız.

Test uzmanının tipik görevleri aşağıdakileri içerebilir:

- Test **planlarını gözden geçirmek** ve planlara **katkı yapmak**
- Gereksinimleri, kullanıcı hikâyelerini ve kabul kriterlerini, spesifikasyonları ve test esasını analiz etmek, gözden geçirmek ve değerlendirmek
- Test koşullarını tanımlanmak ve dokümante etmek ve test senaryoları, test koşulları ve test esası arasındaki **izlenebilirliği kayıt etmek**
- Genellikle sistem yönetimi ve ağ yönetimi ile koordinasyon içinde, test ortamı/ortamlarını tasarlamak, kurmak ve doğrulamak
- Test senaryolarını ve test prosedürlerini tasarlamak ve uyarlamak
- Test verilerini hazırlanmak ve elde etmek
- Ayrıntılı test yürütme çizelgesini oluşturmak
- Testleri koşturmak, sonuçlarını değerlendirmek ve beklenen sonuçlardan sapmaları dokümante etmek
- Test sürecini kolaylaştırmak için uygun araçları kullanmak
- Gerektiğinde testleri otomatikleştirmek (bir yazılımcı veya bir test otomasyon uzmanı tarafından desteklenebilir)
- Performans, güvenilirlik, kullanılabilirlik, güvenlik, uyumluluk ve taşınabilirlik gibi fonksiyonel olmayan testleri yapmak
- Diğerleri tarafından yapılan testleri gözden geçirmek

(Test uzmanının görevleri; test koşullarını tanımlamak, dökümante etmek, user story, test condition ve test base arasındak tracebility'yi kayıt etmek. Test ortamını tasarlamak, kurmak ve doğrulamak. Test prosedürlerini tasarlamak ve uyarlamak. Performans, güvenilirlik, kullanılabilirlik, güvenlik, uyumluluk ve taşınabilirlik gibi fonksiyonel olmayan testleri yapmak)

Test analizi, test tasarımı, farklı test çeşitleri veya test otomasyonu gibi alanlarda çalışan kişiler bu rollerde uzmanlaşabilirler. Ürün ve proje ile ilgili risklere ve seçilen yazılım geliştirme yaşam döngüsü modeline bağlı olarak, farklı kişiler farklı test seviyelerinde test uzmanı rolünü üstlenebilir. Örneğin, birim testleri seviyesinde ve birim entegrasyon testleri seviyesinde, test uzmanının rolü genellikle yazılımcılar tarafından gerçekleştirilir. Kabul testi seviyesinde, test uzmanının rolü genellikle iş analistleri, konunun uzmanları ve kullanıcılar tarafından gerçekleştirilir. Sistem testi seviyesinde ve sistem entegrasyon testi seviyesinde, test uzmanının rolü genellikle bağımsız bir test ekibi tarafından gerçekleştirilir. Operasyonel kabul testi seviyesinde, test uzmanının rolü genellikle operasyon ve/veya sistem yönetim personeli tarafından gerçekleştirilir.

(farklı kişiler farklı test seviyelerinde, test uzmanı rolünü üstlenebilir. Örneğin, Component test developerlar tarafından, acceptance test iş analistleri,konunun uzmanları ve kullanıcılar tarafından gerçekleştirilir, sistem testi bağımsız test ekibi tarafından gerçekleştirilir)

5.2 Test Planlama ve Tahminleme(Test Planning and Estimation, Testplanung und -schätzung)

5.2.1 Test Planının Amacı ve İçeriği

Bir test planı, yazılım geliştirme ve bakım projeleri için **test aktivitelerini özetler.** Planlama; kurumun test politikası ve test stratejisinden, kullanılan yazılım geliştirme yaşam döngülerinden ve yöntemlerinden (bkz. Bölüm 2.1), testlerin kapsamından, hedeflerden, risklerden, kısıtlamalardan, kritiklik durumundan, test edilebilirlik ve kaynakların kullanılabilirliğinden etkilenir.

(Test planı; kurumun test politikası, test stratejisi, kullanılan SDLC modeli, testlerin kapsamı, hedefi, riskler, kısıtlamalar, kritiklik durumu, test edilebilirlik ve kaynakların kullanılabilirliği gibi etmenlere bağlı olarak şekillenir, bakım aşamasını da içerecek şekilde yapılır)

Proje ve test ilerledikçe, daha fazla bilgi elde edilir ve test planına daha fazla ayrıntı eklenebilir. Test planlaması sürekli devam eden bir faaliyettir ve ürünün tüm yaşam döngüsü boyunca gerçekleştirilir. (Ürünün yaşam döngüsünün, bakım aşamasını da içerecek şekilde bir projenin kapsamının ötesine uzanabileceği unutulmamalıdır.) Test aktivitelerinden gelen geri bildirimler, değişen riskleri tanımlamak için kullanılmalı, böylece planlama revize edilmelidir. **Planlama, master test planında ve sistem testleri ve kabul testleri gibi test seviyeleri için veya kullanılabilirlik testleri ve performans testleri gibi farklı test çeşitleri için ayrı test planlarında dokümante edilebilir.**

(Planlama, değişik test çeşitleri, değişik test seviyeleri için ayrı dökümante edilebilir)

Test planlama faaliyetleri aşağıdakileri içerebilir ve bunlardan bazıları bir test planında dokümante edilebilir:

- Testlerin **kapsamı, hedefleri ve risklerinin belirlenmesi**
- Testlerdeki **genel yaklaşımın** belirlenmesi
- Test faaliyetlerinin yazılım **yaşam döngüsü** faaliyetlerine **entegre edilmesi ve koordine edilmesi**
- **Neyin test edileceğine**, çeşitli test faaliyetlerini gerçekleştirmek için **gereken personele** ve diğer kaynaklara ve test faaliyetlerinin nasıl yürütüleceğine ilişkin kararlar verilmesi.
- Test analizi, tasarım, uyarlama, koşturma ve değerlendirme faaliyetlerinin(evaluation activities, Bewertung), belirli tarihlerde (örneğin sıralı yazılım geliştirmede) veya her döngü kapsamında (örneğin döngüsel yazılım geliştirmede) **zaman planlamasının yapılması**
- Test gözetimi ve kontrolü(**test monitoring and control**) için metriklerin seçilmesi
- **Test faaliyetlerinin bütçelendirilmesi**

- Test dokümantasyonunun **detay seviyesinin ve yapısının belirlenmesi** (örneğin, şablonlar veya örnek dokümanlar sağlayarak)

(Test planlama faaliyetleri; Testlerin kapsamı, hedefleri ve risklerinin, genel yaklaşımın belirlenmesi. Test faaliyetlerinin SDLC faaliyetlerine entegre edilmesi. Zaman planlamasının yapılması. Test monitoring ve kontrol için metriklerin seçilmesi. Bütçenin planlanması)

Test planlarının içeriği değişkendir ve yukarıda belirtilen konuların dışına da çıkabilir. Örnek test planları ISO standardında bulunabilir (ISO/IEC/IEEE 29119-3).

5.2.2 Test Stratejisi ve Test Yaklaşımı

Test stratejisi, test sürecinin genellikle ürün veya kurum düzeyinde **genel bir tanımını sağlar.**

Yaygın test stratejisi çeşitleri aşağıda verilmiştir:

- Analitik(Analytical, Analytisch): Bu tür test stratejileri, bazı faktörlerin (örneğin, **gereksinim veya risk) analizine dayanır**. **Risk bazlı testler, testlerin risk seviyesine göre tasarlandığı ve önceliklendirildiği analitik bir yaklaşım örneğidir.**
- Model Bazlı(Model-Based, Modelbasiert): Bu tür test stratejilerinde testler; fonksiyon, iş süreci, iç çalışma yapısı veya fonksiyonel olmayan bir özellik (örneğin güvenilirlik) gibi, **ürünün bazı özelliklerinin modellenmesine dayanarak tasarlanır**. Bu modellerin örnekleri arasında iş süreci modelleri, durum modelleri ve güvenilirlik büyüme modelleri bulunur.
- Metodik(Methodical, Methodisch): Bu tür test stratejileri; **yaygın veya muhtemel arıza türlerinin sınıflandırması**, önemli kalite özelliklerinin listesi veya kurum genelinde mobil uygulamalar veya web sayfaları için görünüm ve çalışma standartları gibi **bazı önceden tanımlanmış test kümelerinin veya test koşullarının sistematik olarak kullanılmasına dayanır.**
- Süreç uyumluluk(Process-compliant, Prozesskonforme) (veya standartlara uyumluluk): Bu tür test stratejileri; **dış yönergeleri ve standartları baz alarak testlerin analiz edilmesini, tasarlanmasını ve uyarlanmasını içerir.** Bu kurallar ve standartlara örnek olarak sektörel standartlar, süreç dokümantasyonu, test esasının titiz bir şekilde tanımlanması ve kullanılması ile belirlenen veya kurumun uygulanmasını zorunlu tuttuğu veya kurumun uygulamasının zorunlu olduğu herhangi bir süreç veya standart olabilir.
- Yönlendirmeli(Directed, Angeleitete) (veya danışılarak): Bu tür test stratejileri, temel olarak test ekibinin dışında veya **kurum dışından paydaşların, alan uzmanlarının veya teknoloji uzmanlarının tavsiyesi, rehberliği veya talimatları ile yönlendirilir.**
- Regresyon hassasiyetli(Regression-averse, Regressionsvermeidend): Bu tür test stratejileri, yazılımın **mevcut çalışan özelliklerinin bozulmasını (regresyon) engelleme amacıyla hayata geçirilmektedir.** Bu test stratejisi, mevcut test yazılımının yeniden kullanımını (özellikle test senaryoları ve test verileri), **geniş regresyon testi otomasyonunu ve standart test gruplarını içerir.**
- Tepkisel(Reactive, Reaktiv): Bu tür test stratejilerinde, testler, (önceki stratejiler gibi) **önceden planlanmak yerine, test edilen birim veya sisteme ve test koşumu sırasında meydana gelen olaylara verilen tepkilerle şekillenir. Testler tasarlanır, uyarlanır ve önceki test sonuçlarından**

elde edilen deneyimler baz alınarak koşturulur. Keşif testleri, tepkisel stratejilerde kullanılan yaygın bir tekniktir.

(Analitik(Analytical, Analytisch): Gereksinim veya risk analizine dayanır. Risk bazlı testler, testlerin risk seviyesine göre tasarlandığı ve önceliklendirildiği analitik bir yaklaşım örneğidir.
Model Bazlı(Model-Based, Modelbasiert): ürünün bazı özelliklerinin modellenmesine dayanarak tasarlanır.
Metodik(Methodical, Methodisch): yaygın veya muhtemel arıza türlerinin sınıflandırması gibi bazı önceden tanımlanmış test kümelerinin veya test koşullarının sistematik olarak kullanılmasına dayanır.
Süreç uyumluluk(Process-compliant, Prozesskonforme) (veya standartlara uyumluluk): dış yönergeleri ve standartları baz alarak testlerin analiz edilmesini, tasarlanmasını ve uyarlanmasını içerir.
Yönlendirmeli(Directed, Angeleitete) (veya danışılarak): Testlerin kurum dışından paydaşların, alan uzmanlarının veya teknoloji uzmanlarının tavsiyesi, rehberliği veya talimatları ile yönlendirilmesine dayalı bir test stratejisidir.
Regresyon hassasiyetli(Regression-averse, Regressionsvermeidend): yazılımın mevcut çalışan özelliklerinin bozulmasını (regresyon) engelleme amacıyla hayata geçirilmektedir. geniş regresyon testi otomasyonunu ve standart test gruplarını içerir.
Tepkisel(Reactive, Reaktiv): önceki stratejiler gibi önceden planlanmak yerine, test edilen birim veya sisteme ve test koşumu sırasında meydana gelen olaylara verilen tepkilerle şekillenir. Testler tasarlanır, uyarlanır ve önceki test sonuçlarından elde edilen deneyimler baz alınarak koşturulur. Keşif testleri, tepkisel stratejilerde kullanılan yaygın bir tekniktir)

Uygun bir test stratejisi genellikle bu tür test stratejilerinin birçoğunun bir araya getirilmesiyle oluşturulur. Örneğin, risk bazlı testler (analitik bir strateji), keşif testleriyle (tepkisel bir strateji) birleştirilebilir; bu ikisi birbirini tamamlar ve birlikte kullanıldığında daha etkili testler elde edilebilir.

Test stratejisi, test sürecinin genel bir tanımını sağlarken, test yaklaşımı belirli bir proje veya sürüm için test stratejisini düzenler. Test yaklaşımı; test tekniklerini, test seviyelerini ve test çeşitlerini seçmek ve giriş kriterlerini ve çıkış kriterlerini (veya sırasıyla Hazır tanımını ve Tamamlandı tanımını) tanımlamak için başlangıç noktasıdır.

Stratejinin uyarlanması; projenin karmaşıklığı ve hedefleri, geliştirilen ürünün türü ve ürün riski analizi ile ilgili olarak verilen kararlara dayanır. Seçilen yaklaşım proje ve test bağlamına bağlıdır ve riskler, emniyet, mevcut kaynaklar ve beceriler, teknoloji, sistemin yapısı (örneğin, ticari paket yazılım), test hedefleri ve düzenlemeler gibi faktörleri dikkate alabilir.

(Test yaklaşımı; test tekniklerini, test seviyelerini ve test çeşitlerini seçmek ve giriş kriterlerini ve çıkış kriterlerini tanımlamak için başlangıç noktasıdır. Test yaklaşımı test stratejisini düzenler bu stratejiye görede test süreci belirlenir.

Projenin karmaşıklığı ve hedefleri, geliştirilen ürünün türü ve ürün riski analizine göre bir Test stratejisi belirlenir)

5.2.3 Giriş Kriterleri ve Çıkış Kriterleri(Entry Criteria and Exit Criteria, Eingangs- und Endekriterien) (Hazır Tanımı ve Tamamlandı Tanımı)

Yazılımın ve testlerin kalitesi üzerinde **etkin bir kontrol sağlamak için**, belirli bir test faaliyetinin **ne zaman başlayacağını ve faaliyetin ne zaman tamamlandığını** tanımlayan kriterlere sahip olunması önerilir. Giriş kriterleri(Entry Criteria) (genellikle çevik yazım geliştirmede "Hazır"(**Ready**) tanımı olarak adlandırılır), **belirli bir test faaliyetinin gerçekleştirilmesi için gereken ön koşulları tanımlar**. **Giriş kriterleri karşılanmazsa, faaliyetin daha zor olması, daha uzun sürmesi, daha maliyetli ve daha riskli olması muhtemeldir**. Çıkış kriterleri(Exit Criteria) (genellikle çevik yazılım geliştirmede "Tamamlandı"(**done**) tanımı olarak adlandırılır), **bir test seviyesi veya bir test grubunu tamamlandı olarak tanımlamak için hangi koşulların sağlanması gerektiğini tanımlar**. **Test hedeflerine göre farklılık gösterebilecek olan giriş ve çıkış kriterleri her test seviyesi ve test çeşidi için tanımlanmalıdır.**

(Giriş kriterleri(Entry Criteria)(Ready): Belirli bir test faaliyetinin gerçekleştirilmesi için gereken ön koşullara denir. Giriş kriterleri karşılanmazsa, faaliyetin daha zor olması, daha uzun sürmesi, daha maliyetli ve daha riskli olma ihtimali vardır.
Çıkış kriterleri(Exit Criteria)(Done): Bir test seviyesi veya bir test grubunu tamamlandı olarak tanımlamak için hangi koşulların sağlanması gerektiğini tanımlar.

Test hedefi değiştikçe Entry ve Exit Criterialarda değişir. Her test seviyesi ve her test çeşidi için Entry ve Exit Criteria tanımlanması gerekiyor)

Tipik giriş kriterleri aşağıda verilmiştir:

- Test edilebilir gereksinimlerin, kullanıcı hikâyelerinin ve/veya modellerin var olması (örneğin, model bazlı bir test stratejisi izlenirken)
- Önceki test seviyelerinin çıkış kriterlerini karşılayan test öğelerinin varlığı
- Test ortamının kullanılabilir olması
- Gerekli test araçlarının kullanılabilir olması
- Test verilerinin ve diğer gerekli kaynakların varlığı

Tipik çıkış kriterleri aşağıda verilmiştir:

- Planlanan testlerin koşturulmuş olması
- Belirlenmiş bir kapsama seviyesine (örneğin, gereksinimler, kullanıcı hikâyeleri, kabul kriterleri, riskler, kod) ulaşılmış olması
- Çözülemeyen hataların sayısının önceden kararlaştırılmış bir limitin dahilinde olması
- Tahmini kalan hata sayısının yeterince düşük olması
- Değerlendirilen güvenilirlik seviyeleri, performans, kullanılabilirlik, güvenlik ve diğer ilgili kalite özelliklerinin yeterli olması

Çıkış kriterleri karşılanmasa bile, bütçenin tükenmesi, planlanan süresinin tamamlanması ve/veya ürünün piyasaya sürülmesi için oluşan baskı nedeniyle test faaliyetlerinin kısa kesilmesi de yaygın bir durumdur.

Proje paydaşları ve ürün sahipleri daha fazla test yapmadan ürünü piyasaya sürmenin riskini gözden geçirip kabul ettiyse, bu şartlar altında testlere son verilmesi kabul edilebilir bir durumdur.

(Bütçenin tükenmesi, planlanan sürenin tamamlanması, ürünün piyasaya sürülmesi için oluşan baskı gibi nedenlerle, proje paydaşları ve ürün sahipleri daha fazla test yapmadan ürünü piyasaya sürmenin riskini gözden geçirip kabul ettiyse, çıkış kriterleri sağlanmasa bile testlere son verilir)

5.2.4 Test Koşum Çizelgesi(Test Execution Schedule, Testausführungsplan)

Çeşitli test senaryoları ve test prosedürleri (potansiyel olarak bazı test prosedürleri otomatikleştirilmiş olabilir) üretildikten ve test gruplarına ayrıldıktan sonra, test grupları, çalıştırılma sırasını belirleyen bir test koşum çizelgesinde düzenlenebilir. Test koşum çizelgesi(Test Execution Schedule, Testausführungsplan); önceliklendirme(prioritization), bağımlılıklar(dependencies, Abhängigkeiten) onaylama testleri(confirmation tests, Fehlernachtests), regresyon testleri ve testleri koşturmak için en verimli sıralama gibi faktörleri dikkate almalıdır.

(User Story'ler ve Test prosedürleri oluşturulur Bunlar test gruplarına ayrılır sonra çalıştırma sırasına göre bir Test execution Schedule oluşturulur.
Test koşum çizelgesinde(Test Execution Schedule, Testausführungsplan); önceliklendirme(prioritization), bağımlılıklar(dependencies, Abhängigkeiten) onaylama testleri(confirmation tests, Fehlernachtests), regresyon testleri ve testleri koşturmak için en verimli sıralama gibi faktörler dikkate alınarak bir sıralama oluşturulur)

İdeal olarak, test senaryoları öncelik seviyelerine göre çalışma sırasına alınır, genellikle en yüksek önceliğe sahip test senaryoları önce koşturulur. Ancak, test senaryolarının veya test edilen özelliklerin bağımlılıkları varsa, bu yöntem çalışmayabilir.

Yüksek öncelikli bir test senaryosu düşük öncelikli bir test senaryosuna bağımlıysa, önce düşük öncelikli test senaryosu koşturulmalıdır.

Benzer şekilde, test senaryoları arasında bağımlılıklar varsa, göreceli öncelikleri ne olursa olsun uygun şekilde sıralama yapılmalıdır. Değişikliklerle ilgili hızlı geri bildirimin önemine dayanarak, onaylama ve regresyon testlerine de öncelik verilmelidir, ancak yine burada da bağımlılıklar geçerli olabilir.

Bazı durumlarda birçok test sıralaması mümkündür; bu sıralamaların verimlilik seviyeleri de farklılık gösterir. Bu gibi durumlarda, test koşum verimliliği ile önceliklendirme arasında bir denge kurulmalıdır.

5.2.5 Test Eforunu(Test Effort, Testaufwand) Etkileyen Faktörler

Test eforu(Test Effort, Testaufwand) tahminlemesi, belirli bir proje, sürüm veya döngüdeki testlerin hedeflerini yerine getirmek için **yapılması gereken testlerle ilgili çalışmanın miktarını öngörmektir**. Test çalışmasını etkileyen faktörler arasında ürünün özellikleri, yazılım geliştirme sürecinin özellikleri, testi yapacak kişilerin özellikleri ve test sonuçları sayılabilir. Bunlar aşağıda verilmiştir.

(Bir proje istenilen hedeflere ulaşılabilecek mi, istenildiği gibi çalışacak mı, bunun doğru ve sağlıklı bir şekilde tespitini yapabilmek için bu projeye uygulanması gereken tahmini test sayısına test eforu diyoruz Test çalışmasını etkileyen faktörler; ürünün özellikleri, yazılım geliştirme sürecinin özellikleri, testi yapacak kişilerin özellikleri ve test sonuçları sayılabilir)

Ürünün özellikleri
- Ürünle ilgili riskler
- Test esasının kalitesi
- Ürün boyutu
- Ürünün karmaşıklığı
- Kalite gereksinimleri (ör. güvenlik, güvenilirlik)
- Test dokümantasyonu için gereken ayrıntı düzeyi
- Yasal ve düzenlemeler için uyumluluk gereksinimleri

Yazılım geliştirme süreci özellikleri
- Kurumun kararlılığı ve olgunluğu
- Kullanılan yazılım geliştirme modeli
- Test yaklaşımı
- Kullanılan araçlar
- Test süreci
- Zaman baskısı

Testi Yapacak Kişilerin özellikleri
- Kişilerin becerileri ve özellikle benzer projeler ve ürünlere (ör. alan bilgisi) ilişkin deneyimleri
- Ekip uyumu ve liderlik

Test sonuçları
- Belirlenen hataların sayısı ve önem derecesi
- Yeniden yapılması gereken iş miktarı

5.2.6 Test Tahminleme Teknikleri(Test Estimation Techniques, Testschätzverfahren)

Yeterince test etmek için **gerekli olan eforu belirlemek** amacıyla kullanılan **bazı tahminleme teknikleri vardır**. En sık kullanılan tekniklerden ikisi aşağıda verilmiştir:

*(Ne kadar test yapmak gerektiğini belirlemek(Test eforu) için iki **Test Tahminleme tekniği(Test Estimation Techniques, Testschätzverfahren)** kullanılır)*
- ***Metrik bazlı teknik(metrics-based technique, metrikbasierte Verfahren):** önceki benzer projelerin metriklerine veya tipik değerlere dayanarak test eforunu tahmin etme*
- ***Uzman bazlı teknik(expert-based technique, expertenbasierte Verfahren):** testi gerçekleştirecek kişilerin tecrübesine veya uzmanlara dayanarak test eforunu tahmin etme*

Örneğin, çevik yazılım geliştirmede, burndown tabloları (yapılacak işler tablosu), metrik bazlı yaklaşımın örneklerindendir; çalışma eforu bu tablolara kaydedilip raporlanır ve daha sonra ekibin bir sonraki döngüde

yapabileceği iş miktarını belirlemek için ekibin hızını düzenlemekte kullanılır; poker planlama (Wideband Delphi tahminleme tekniğini baz alır) ise, ekip üyelerinin kendi deneyimlerine dayanarak yapılacak iş için gereken çalışmayı tahmin ettikleri bir tekniktir (ISTQB-AT Temel Seviye Çevik Test Uzmanı Devam Kursu Ders Programı).

Agile yazılım geliştirmede (Burndown tabloları (yapılacak işler tablosu), metrik bazlı yaklaşımın örneklerindendir. çalışma eforu bu tablolara kaydedilip raporlanır ve daha sonra ekibin bir sonraki döngüde yapabileceği iş miktarını belirlemek için ekibin hızını düzenlemekte kullanılır. Waterfoll gibi Sıralı(sequential) metodolojileri kullanan projelerde, hata giderme modelleri metrik bazlı yaklaşımın örnekleridir.
poker planlama (Wideband Delphi tahminleme tekniğini baz alır) ise, ekip üyelerinin kendi deneyimlerine dayanarak yapılacak iş için gereken çalışmayı tahmin ettikleri bir tekniktir.
Wideband Delphi tahminleme tekniği ise, uzman gruplarının kendi deneyimlerine dayanarak tahminler sunduğu uzman bazlı yaklaşımın bir örneğidir)

Şelale yazılım geliştirme metodolojisi gibi sıralı(sequential) metodolojileri kullanan projelerde, hata giderme modelleri metrik bazlı yaklaşımın örnekleridir; burada bulunan hatalar, sayıları ve bunları düzeltmek için gereken süre belirlenip raporlanır, bu da benzer nitelikteki gelecek projelerde tahminleme için bir esas oluşturur. Wideband Delphi tahminleme tekniği ise, uzman gruplarının kendi deneyimlerine dayanarak tahminler sunduğu uzman bazlı yaklaşımın bir örneğidir (ISTQB-ATM İleri Seviye Test Yöneticisi Ders Programı).

5.3 Test Gözetimi ve Kontrolü(Test Monitoring and Control)

Test gözetiminin(test monitoring, Testüberwachung) amacı, **test faaliyetleri hakkında bilgi toplamak** ve **geri bildirim ve görünürlük(visibility) sağlamaktır**. **İzlenecek bilgiler** manuel veya otomatik olarak toplanabilir ve test ilerlemesini değerlendirmek ve test çıkış kriterlerinin veya ürün risklerinin kapsanması, gereksinimler veya kabul kriterlerinin karşılanması için hedeflerin tutturulması gibi bir çevik projenin "Tamamlandı" tanımıyla ilgili **test görevlerinin yerine getirilip getirilmediğini ölçmek için kullanılmalıdır.**

Test kontrolü, toplanan ve (muhtemelen) raporlanan bilgilerin ve metriklerin bir sonucu olarak gerçekleştirilen **yönlendirici veya düzeltici eylemleri tanımlar**. Eylemler herhangi bir test faaliyetini içerebilir ve diğer yazılım yaşam döngüsü faaliyetlerini etkileyebilir.

(Test gözetiminin(test monitoring, Testüberwachung) amacı, test faaliyetleri hakkında bilgi toplamak ve geri bildirim ve görünürlük(visibility) sağlamaktır. İzlenecek bilgiler test görevlerinin yerine getirilip getirilmediğini ölçmek için kullanılmalır.
Test kontrolü(Test control, Teststeurung), toplanan ve (muhtemelen) raporlanan bilgilerin ve metriklerin bir sonucu olarak gerçekleştirilen yönlendirici veya düzeltici eylemleri tanımlar)

Test kontrol eylemlerine örnekler aşağıda sıralanmıştır:

- **Tanımlanmış bir risk** (örneğin, geç teslim edilen yazılım) gerçekleştiğinde testlerin yeniden önceliklendirilmesi
- Test ortamının veya diğer kaynakların kullanılabilir veya kullanılamaz olması nedeniyle test zaman çizelgesinin değiştirilmesi
- Bir test öğesinin yeniden ele alınması sonucunda bir giriş veya çıkış kriterini karşılayıp karşılamadığının yeniden değerlendirilmesi

5.3.1 Yazılım Testlerinde Kullanılan Metrikler(ölçüm)

Aşağıdakileri değerlendirmek için test faaliyetleri sırasında ve sonunda metrikler(ölçüm) toplanabilir:

- Planlanan zaman çizelgesine ve bütçeye göre ilerleme durumu
- Test nesnesinin mevcut kalitesi
- Test yaklaşımının uygunluğu
- Test faaliyetlerinin hedeflere göre etkinliği

Yaygın test metrikleri aşağıdakileri içerir:

- **Test senaryosu** hazırlamada **yapılan planlı işlerin yüzdesi** (veya koşulan planlı test senaryolarının yüzdesi)
- **Test ortamı** hazırlamada **yapılan planlı işlerin yüzdesi**
- Test senaryosunun **koşturulması** (örneğin, çalıştırılmış/çalıştırılmamış test senaryolarının sayısı, başarılı/başarısız test senaryoları ve/veya başarılı/başarısız test koşulları)
- **Hata bilgileri** (örneğin hata yoğunluğu, belirlenen ve çözülen hatalar, arıza oranı ve onaylama testi sonuçları)

- Gereksinimlerin, kullanıcı hikâyelerinin, kabul kriterlerinin, risklerin veya kodun **test kapsamı**
- **Görev tamamlama, kaynak tahsisi, kaynak kullanımı ve çalışma**
- **Maliyet/hata** bulma faydasının karşılaştırılması veya **maliyet/testi** çalıştırma faydasının karşılaştırılması da dâhil olmak üzere **testlerin maliyeti**

5.3.1 Test Raporlarının Amaçları, İçeriği ve Hedef Kitlesi(Purposes, Contents, and Audiences for Test Reports, Zwecke, Inhalte und Zielgruppen für Testberichte)

Test raporlamanın amacı, bir test faaliyeti sırasında ve sonunda (örneğin bir test seviyesi) test faaliyeti bilgilerini özetlemek ve iletmektir.

***Test faaliyeti(test activity,Testaktivität)** sırasında hazırlanan test raporuna **test ilerleme raporu(test progress report, Testfortschrittsbericht)**, test faaliyeti sonunda hazırlanan test raporuna ise **test özet raporu(test summary report, Testabschlussbericht)** denebilir.*

Test gözetimi ve kontrolü sırasında, **test yöneticisi** düzenli olarak paydaşlar için **test ilerleme raporları hazırlar**. Test ilerleme raporları ve test özet raporları için ortak içeriğe ek olarak, **tipik test ilerleme raporları aşağıdakileri de içerebilir:**

- Test planına göre test **faaliyetlerinin durumu ve ilerlemesi**
- İlerlemeyi engelleyen faktörler
- Sıradaki raporlama dönemi için planlanan testler
- Test **nesnesinin kalitesi**

(Test ilerleme raporları; faaliyetlerinin durumu ve ilerlemesi, ilerlemeyi engelleyen faktörler, sıradaki planlanan testler, Test objesinin kalitesi gibi bilgiler içerir. Bu rapor exit criteria'ya ulaşıldığında Testmanager tarafından oluşturulur.)

Çıkış kriterlerine ulaşıldığında, **test yöneticisi test özet raporunu oluşturur.** Bu rapor, en son test ilerleme raporuna ve diğer ilgili bilgilere dayanarak yapılan testlerin özeti niteliğindedir.

Test ilerleme raporları ve test özet raporları genellikle aşağıdakileri içerir:

- Gerçekleştirilen **testlerin özeti**
- **Bir test dönemi sırasında neler oldu**ğuna dair bilgiler
- Test faaliyetlerinin zaman çizelgesindeki, **süresindeki veya çalışma miktarındaki sapmalar da dâhil olmak üzere plandan sapmalar**
- Çıkış kriterleri veya Tamamlandı tanımına göre **testlerin ve ürün kalitesinin durumu**
- **İlerlemeyi engellemiş veya engellemeye devam eden faktörler**
- **Hata metrikleri, test senaryoları, test kapsamı, faaliyet ilerlemesi ve kaynak tüketimi**. (ör. 5.3.1'de açıklandığı gibi)
- **Kalan riskler** (bkz bölüm 5.5)
- Üretilen yeniden kullanılabilir **test çalışma ürünleri**

(Test özet raporları; testlerin özeti, test aktiviteleri sırasında karşılaşılanlar, planlanan sürede ve çalışma miktarındaki sapmalar, testlerin ve ürün kalitesinin durumu, İlerlemeyi engellemiş veya engellemeye devam eden faktörler, hata metrikleri, test kapsamı, kaynak tüketimi, kalan riskler, yeni üretilen test çalışma ürünleri. Test raporunun içeriği projeye, organizasyonel gereksinimlere ve yazılım geliştirme yaşam döngüsüne bağlı olarak değişir. Test lead tarafından oluşturulur)

Test raporunun içeriği projeye, organizasyonel gereksinimlere ve yazılım geliştirme yaşam döngüsüne bağlı olarak değişecektir. Örneğin, birçok paydaşın olduğu karmaşık bir proje veya düzenlemelere tabi bir proje, hızlı bir yazılım güncellemesinden daha ayrıntılı ve titiz bir raporlama gerektirebilir. Diğer bir örnek olarak, çevik yazılım geliştirmede test ilerleme raporunun hazırlanması, görev panolarına, hata özetlerine ve yapılacak işler (burndown) tablolarına dâhil edilebilir ve günlük ayakta toplantılarda(daily stand-up meeting, täglichen Stand-up-Meeting besprochen) tartışılabilir (bkz. ISTQB-AT Temel Seviye Çevik Test Uzmanı Devam Kursu Ders Programı).

Test raporlarını projenin bağlamına göre uyarlanmanın yanı sıra rapor hedef kitlesine göre de düzenlenmelidir. **Teknik bir hedef kitleye veya bir test ekibine yönelik verilen bilgilerin türü ve miktarı, yönetici özet raporunda verilenden farklı olabilir.** İlk durumda, **hata çeşitleri ve eğilimleri hakkında ayrıntılı bilgiler önemli olabilir.** İkinci durumda, üst düzey bir rapor (ör. öncelik sırasına göre hataların, bütçenin, zaman çizelgesinin ve başarılı/başarısız/test edilmemiş test koşullarının durum özeti) daha uygun olabilir.
ISO standardı (ISO/IEC/IEEE 29119-3) test ilerleme raporları ve test tamamlama raporları (bu ders programında test özet raporları olarak adlandırılmıştır) olmak üzere iki tür test raporundan bahseder ve iki rapor türü için de ana başlıklar ve örnekler içerir.

5.4 Yapılandırma Yönetimi(Configuration Management, Konfigurationsmanagement)

Yapılandırma yönetiminin (Configuration Management, Konfigurationsmanagement) *amacı, birim veya sistemin, test yazılımının, proje ve ürün yaşam döngüsü boyunca birbirleriyle olan ilişkilerinin bütünlüğünü sağlamak ve korumaktır.*

Testleri uygun şekilde desteklemek için **yapılandırma yönetimi aşağıdakilerin gerçekleştirilmesini içerebilir**:

- Tüm test öğeleri özgün bir şekilde **tanımlanmış**, **versiyon kontrolü yapılmış**, **değişiklikleri izlenmiş ve birbirleriyle ilişkilendirilmiştir.**
- Test için yazılan tüm yazılımlar benzersiz bir şekilde tanımlanmış, versiyon kontrolü yapılmış, birbirleriyle ve test öğesinin/öğelerinin versiyonlarıyla ilgili değişiklikler için izlenmiş, böylece test süreci boyunca izlenebilirlikleri korunmuştur.
- Tanımlanan tüm dokümanlar ve yazılım öğeleri, **test dokümantasyonunda açık bir şekilde belirtilmiştir**. **Test planlama sırasında yapılandırma yönetimi prosedürleri ve altyapısı (araçlar) tanımlanmalı ve uyarlanmalıdır.**

(Versiyon kontrolü yapma, değişiklikleri izleme, izlenebilirliği koruma Configuration management faaliyetleridir.

Test planlama sırasında yapılandırma yönetimi(Configuration Management) prosedürleri ve altyapısı (araçlar) tanımlanmalı ve uyarlanmalıdır)

5.5 Riskler ve Testler(Risks and Testing , Risiken und Testen)

5.5.1 Risk Tanımı

Risk, gelecekte olumsuz sonuçlara yol açacak bir olayın gerçekleşme olasılığını içerir. Risk seviyesi, olayın olasılığı ve etkisi (zararı) ile belirlenir.

5.5.2 Ürün ve Proje Riskleri

Ürün riski, bir çalışma ürününün (örneğin bir gereksinim, birim, sistem veya test) **kullanıcılarının ve/veya paydaşlarının meşru ihtiyaçlarını karşılamaması ihtimalini içerir.** Ürün riskleri, bir ürünün belirli kalite karakteristikleriyle ilişkilendirildiğinde (örneğin, fonksiyonalite, güvenilirlik, performans, kullanılabilirlik, güvenlik, uyumluluk, sürdürülebilirlik ve taşınabilirlik), **ürün risklerine kalite riskleri de denir.** Ürün risklerine örnekler aşağıda verilmiştir:

- Yazılım, **gereksinime göre amaçlanan fonksiyonları yerine getiremeyebilir**.
- Yazılım; kullanıcı, müşteri ve/veya paydaş ihtiyacına göre **amaçlanan fonksiyonları yerine getiremeyebilir**
- Sistem mimarisi, **fonksiyonel olmayan bazı gereksinimleri** uygun şekilde **desteklemeyebilir**
- Bazı durumlarda belirli bir **hesaplama yanlış** yapılabilir.
- Bir döngü kontrol yapısı **yanlış kodlanmış** olabilir.
- **Yüksek performanslı olması beklenen bir işlemin yanıt süreleri yetersiz olabilir**.
- Kullanıcı deneyimiyle ilgili geri bildirimler, ürün beklentilerini karşılamayabilir.

(Ürün riski, bir çalışma ürününün kullanıcılarının ve/veya paydaşlarının meşru ihtiyaçlarını karşılamaması ihtimalini içerir. Ürün risklerine kalite riskleri de denir(örneğin, fonksiyonalite, güvenilirlik, performans, kullanılabilirlik, güvenlik, uyumluluk, sürdürülebilirlik ve taşınabilirlik gibi ihtiyaçların karşılanmama ihtimali)

Proje riski, gerçekleşmesi durumunda, projenin hedeflerine ulaşma imkânı üzerinde olumsuz etkiye sahip olabilecek durumları içerir.

Proje risklerine örnekler aşağıda verilmiştir:

- Proje sorunları:
  - **Teslimatta**, görevi tamamlamada veya çıkış kriterlerinin veya tamamlandı tanımının karşılanmasında **gecikmeler** olabilir
  - **Yanlış tahminler**, maddi kaynakların daha yüksek öncelikli projelere yeniden tahsis edilmesi veya kurum genelinde genel harcama kısıntıları, parasal kaynak yetersizliğine neden olabilir
  - **Geç yapılan değişiklikler** önemli ölçüde yeniden yapılması gereken işlere neden olabilir

- Kurumsal sorunlar:
  - **Yetkinlik**, eğitim ve personel yeterli olmayabilir
  - Personel sorunları **çatışma** ve problemlere neden olabilir
  - Çakışan işletme öncelikleri nedeniyle kullanıcılara, işletme personeline veya konunun uzmanlarına **ulaşılamayabilir**
- Politik sorunlar:
  - Test uzmanları ihtiyaçlarını ve/veya **test sonuçlarını uygun şekilde iletmeyebilir**
  - Yazılımcılar ve/veya test uzmanları testlerde ve gözden geçirmelerde bulunan bilgilerin **takibini yapmayabilir** (ör. yazılım geliştirme ve test uygulamalarını iyileştirmemek)
  - Testlere ilişkin yanlış bir tutum veya yanlış beklentiler olabilir (örneğin, testler sırasında **hataların belirlenmesinin önemini takdir etmemek**)
- Teknik sorunlar:
  - **Gereksinimler yeterince iyi tanımlanmamış** olabilir
  - Mevcut kısıtlamalar göz önüne alındığında **gereksinimler yerine getirilememiş** olabilir
  - **Test ortamı zamanında hazır olmayabilir**
  - Veri dönüşümü, taşıma planlaması ve bunların araç desteği geç kalmış olabilir
  - **Yazılım geliştirme sürecindeki zayıflıklar**; tasarım, kod, yapılandırma, test verileri ve test senaryoları gibi proje çalışma ürünlerinin tutarlılığını veya kalitesini etkileyebilir
  - Yetersiz hata yönetimi ve benzer problemler, hata birikimi ve diğer teknik borçlarla sonuçlanabilir
- Tedarikçi sorunları:
  - Tedarikçiler, **gerekli bir ürünü** veya hizmeti **sağlayamayabilir** veya iflas edebilir
  - **Sözleşmeden kaynaklanan sorunlar** projede aksaklıklara neden olabilir

Proje riskleri, hem yazılım geliştirme faaliyetlerini hem de test faaliyetlerini etkileyebilir. Bazı durumlarda proje yöneticileri tüm proje riskleriyle ilgilenmekten sorumludur, ancak test yöneticilerinin testle ilgili proje risklerinin sorumluluğunu yüklenmesi de olağandışı değildir.

(Proje risklerine örnekler;
Proje sorunları olarak teslimattaki gecikmeler(yazılımın yavaş çalışması bir proje sorunudur), yanlış tahminler(maddi sorunlara yol açabilir), geç yapılan değişiklikler(bazı işlerin tekrardan yapılması gibi sorunlara yol açabilir).
Kurumsal sorunlar olarak yetkinlik, personel arası çatışmalar, kullanıcı, işletme veya konunun uzmanlarına ulaşılamama.
Politik sorunlar olarak test sonuçlarının sağlıklı iletilmemesi, developerlar veya testerlar bulunan verilerin takibini iyi yapmaması, hataların belirlenmesinin önemini takdir etmeme(hassas olmama).
Teknik sorunlar olarak requirementların iyi tanımlanmaması, test ortamının zamanında hazır olmaması, yazılımın iyi yapılmamış olması.
Tedarikçi sorunları olarak ürünün tedarikçiler tarafından sağlanamaması, sözleşmeden kaynaklanan sorunlar.

Proje riskleri, hem yazılım geliştirme faaliyetlerini hem de test faaliyetlerini etkileyebilir. Bazı durumlarda proje yöneticileri tüm proje riskleriyle ilgilenmekten sorumludur, bazen test managerlarda bu işi yapabilir)

5.5.3 Risk Bazlı(riske dayalı) Testler ve Ürün Kalitesi

Risk, **test eforunu odaklamak** için kullanılır. Testlere **nerede ve ne zaman başlanacağını ve daha fazla dikkat gerektiren alanları** belirlemek için kullanılır. Testler, olumsuz bir olay gerçekleşme olasılığını azaltmak veya olumsuz bir olayın etkilerini azaltmak için kullanılır.

(Risk, Testlere nerede ve ne zaman başlanacağını ve daha fazla dikkat gerektiren alanları belirlemek için kullanılır)
Testler, belirlenen risklerin riskini azaltma ve onlar hakkında geri bildirim sağlama, kalan (ele alınamamış) riskler hakkında ise geri bildirim sağlamak için kullanılır.

Testlere yönelik **risk bazlı bir yaklaşım**, ürün riski düzeylerini azaltmak için proaktif fırsatlar sağlar.
Ürün risk analizi, ürün risklerinin belirlenmesi, belirlenen her bir risk için gerçekleşme olasılığı ve etkilerinin değerlendirmesi aktivitelerini içerir. Sonuç olarak elde edilen ürün risk bilgileri; test planlamasını, test gereksinimlerini, hazırlıkları ve test senaryolarının koşturulmasını yönlendirmek ve test gözetimi ve kontrolü için kullanılır. Ürün risklerini erkenden analiz etmek projenin başarısına katkıda bulunur.
Risk bazlı bir yaklaşımda ürün risk analizinin sonuçları aşağıdaki amaçlar için kullanılır:

- Kullanılacak **test tekniklerinin belirlenmesi**
- Koşulacak **testlerin seviyelerinin ve çeşitlerinin belirlenmesi** (ör. güvenlik testleri, erişilebilirlik testleri)
- Koşulacak **testlerin kapsamının belirlenmesi**
- Kritik hataların mümkün olduğunca erken belirlenmesi için **testlerin önceliklendirilmesi**
- Riski azaltmak için testlere ek olarak yapılabilecek potansiyel aktivitelerin belirlenmesi (ör. deneyimsiz tasarımcılara eğitim verilmesi)

(Risk bazlı bir yaklaşımda ürün risk analizinin sonuçları; test tekniklerinin belirlenmesi, testlerin seviyelerinin ve çeşitlerinin belirlenmesi, testlerin kapsamının belirlenmesi, testlerin önceliklendirilmesi, Riski azaltmak için testlere ek olarak yapılabilecek potansiyel aktivitelerin belirlenmesi)

Risk bazlı testler, ürün risk analizini gerçekleştirmek için proje paydaşlarının **ortak bilgi ve sezgilerine** dayanır. Üründe arıza olasılığının en aza indirilmesi için risk yönetimi faaliyetleri disiplinli bir yaklaşım ile aşağıdakileri sağlar:

- Neyin yanlış gidebileceğinin (riskler) analizi (ve düzenli olarak yeniden değerlendirilmesi)
- Hangi risklerin önlenmesinin önemli olduğunun belirlenmesi
- Bu riskleri azaltmak için aksiyon alınması
- Gerçekleşmeleri halinde risklerle başa çıkmak için acil durum planlarının yapılması

(Üründe arıza(failures) olasılığının en aza indirilmesi için; risk analizi yapılması, risklerin önceliklendirilmesi, bu riskleri azaltmak için gereken aktivitelerin çıkarılması, risklerin ortaya çıkması durumunda acil durum planı yapılması)

Buna ek olarak, testler yeni riskleri belirleyebilir, hangi risklerin azaltılması gerektiğinin belirlenmesine yardımcı olabilir ve risklerle ilgili belirsizliği azaltabilir.

5.6. Hata Yönetimi(Defect Management, Fehlermanagement)

Testlerin hedeflerinden biri de hataların bulunması olduğu için **testlerde bulunan hatalar raporlanmalı ve kaydedilmelidir**.Hataların **kaydedilme şekli, test edilen birim veya sistemin bağlamına, test seviyesine ve yazılım geliştirme yaşam döngüsü modeline bağlı olarak değişebilir**. Bulunan her hata araştırılmalı ve **bulunma aşamasından, sınıflandırılmasından, düzeltilmesine kadar takip edilmelidir** (ör. Hataların düzeltilmesi, hata çözümü üzerinde onaylama testlerinin başarıyla uygulanması, hatanın sonraki bir sürüme ertelenmesi, kalıcı bir ürün sorunu olarak kabul edilmesi gibi). **Kurum**, bütün hataların düzeltilme sürecini yönetmek için, **iş akışı ve sınıflandırma kurallarını içeren bir hata yönetim süreci oluşturmalıdır.** Bu süreç; tasarımcılar, yazılımcılar, test uzmanları ve ürün sahipleri de dâhil olmak üzere hata yönetimindeki tüm paydaşlar tarafından kabul edilmelidir. Bazı kurumlarda hata kaydı ve izlenmesi oldukça gayri resmi olabilir.

(Testlerde bulunan hatalar raporlanmalı ve kaydedilmelidir. Kaydedilme şekli birim veya sistem bağlamına, test level'ına ve SDLC modeline göre değişir. Hatalar bulunma aşamasından, sınıflandırma ve düzeltilmesine kadar takip edilmelidir)

Hata yönetimi sürecinde bazen hatalardan kaynaklanan gerçek arızalar değil yanlış hatalar, yanlış pozitifleri raporlanabilir. Örneğin, bir ağ bağlantısı kesildiğinde veya süresi dolduğunda bir test başarısız olabilir. Bu başarısızlık, test nesnesindeki bir hatadan kaynaklanmamaktadır, ancak araştırılması gereken bir anomalidir. Test uzmanları, hata olarak bildirilen yanlış pozitiflerin sayısını azaltmaya çalışmalıdır.
Kodlama, statik analiz, gözden geçirmeler, dinamik testler veya yazılım ürününün kullanımı sırasında hatalar bulunabilir. Gereksinimler, kullanıcı hikâyeleri ve kabul kriterleri, yazılım geliştirme dokümanları, test dokümanları, kullanıcı kılavuzları veya kurulum kılavuzları dâhil olmak üzere herhangi bir dokümantasyon türü içindeki, koddaki veya sistemdeki sorunlar için hatalar bildirilebilir.

Kurumlar, etkin ve verimli bir hata yönetimi sürecine sahip olmak amacıyla hataların nitelikleri, sınıflandırması ve iş akışı için standartlar tanımlayabilir.

Arıza(Defect) raporlarının amaçları genellikle aşağıdaki gibidir:

- Gerçekleşen herhangi bir **olumsuz olayın etkilerini tanımlamak**, **arızayı yeniden oluşturabilmek için gereken adımları belirlemek ve hataları düzeltmek için yazılımcılara ve diğer paydaşlara gereken bilgileri sunmak**
- **Test yöneticilerine, çalışma ürününün kalitesini ve testler üzerindeki etkisini izleme imkânı sağlamak** (örneğin çok fazla hata bulunursa test uzmanları, testleri koşmak yerine bunları raporlamak için çok zaman harcayacaktır ve daha fazla onaylama testi gerekecektir)
- **Yazılım geliştirmenin ve test sürecinin iyileştirilmesi konusunda fikirler sunmak**

Dinamik testler sırasında oluşturulan bir hata raporu genellikle aşağıdakileri içerir:

- Hatanın seri numarası veya sırası
- Bulunan hatanın başlığı ve kısa bir özeti
- Hata raporunun tarihi, düzenleyen kurum ve yazan kişi
- Test öğesinin (test edilen yapılandırma öğesi) ve ortamının tanımı
- Hatanın bulunduğu yazılım geliştirme yaşam döngüsü aşaması/aşamaları

- Günlükler, veri tabanı ekran görüntüleri veya kayıtlar (test koşumu sırasında bulunursa) dâhil olmak üzere hatanın yeniden oluşturulmasına ve çözülmesine olanak sağlamak için hatanın tanımı
- Beklenen ve gerçekleşen sonuçlar
- Hatanın kapsamı veya paydaşlar üzerindeki etki derecesi (önem derecesi)
- Düzeltilme aciliyeti/önceliği
- Hata raporunun durumu (ör. açık, ertelenmiş, tekrarlanmış, düzeltilmeyi bekliyor, onaylama testleri bekleniyor, yeniden açıldı, kapatıldı)
- Sonuçlar, öneriler ve onaylar
- Hatadan kaynaklanan bir değişiklikten etkilenebilecek diğer alanlar gibi genel sorunlar
- Değişiklik geçmişi: ör. proje ekibi üyeleri tarafından gerçekleştirilen, hatayı bulma, giderme ve giderildiğini doğrulama işlemleri dizisi
- Problemi ortaya çıkaran test senaryosu da dâhil olmak üzere verilen referanslar

Bu detaylardan bazıları, hata yönetimi araçlarını kullanırken otomatik olarak üretilebilir; ör. bir seri numarasının otomatik olarak hataya atanması, iş akışı sırasında hata raporu durumunun atanması ve güncellenmesi gibi.

Statik testler, özellikle de gözden geçirmeler sırasında bulunan hatalar, normal olarak farklı bir şekilde dokümante edilir, ör. gözden geçirme toplantısı(review meeting) notlarında. (hata raporları(defect report) olay raporları(incedent report) olarakta adlandırılır ve tester tarafından raporlanır)

Örnek bir hata raporu içeriği, ISO standardında bulunabilir (ISO/IEC/IEEE 29119-3)

CHAPTER 6

6. Yazılım Test Araç Desteği

Anahtar kelimeler

veri güdümlü testler, aksiyon kelimesi güdümlü testler, performans testi aracı, test otomasyonu, test koşum aracı, test yönetim aracı

Keywords
data-driven testing, keyword-driven testing, test automation, test execution tool, test management tool

Schlüsselbegriffe
Datengetriebenes Testen, schlüsselwortgetriebener Test, Testautomatisierung, Testausführungswerkzeug, Testmanagementwerkzeug

6.1 Test Araçlarındaki Önemli Hususlar

Test araçları, bir veya daha fazla test faaliyetini desteklemek için kullanılabilir. Bu araçlar aşağıda verilmiştir:

- Doğrudan testlerde kullanılan araçlar: test koşum araçları ve test verisi hazırlama araçları gibi
- Gereksinimlerin, test senaryolarının, test prosedürlerinin, otomatikleştirilmiş test betiklerinin, test sonuçlarının, test verilerinin ve hataların yönetilmesine yardımcı olan ve test koşumunun raporlanması ve gözetimi için kullanılan araçlar
- Araştırma ve değerlendirme için kullanılan araçlar
- Testlerde yardımcı olan herhangi bir araç (bu anlamda Excel de bir test aracıdır)

6.1.1 Test Aracı Sınıflandırması

Test araçları, proje ve test bağlamına bağlı olarak aşağıdaki amaçlardan bir veya daha fazlasına sahip olabilir:

- Tekrarlayan veya manuel olarak yapıldığında **yüksek miktarda kaynak gerektiren işleri** otomatikleştirerek test faaliyetlerinin verimliliğini artırmak (ör. test koşumu, regresyon testleri)
- Test süreci boyunca manuel test faaliyetlerini destekleyerek test faaliyetlerinin **verimliliğini artırmak** (bkz. Bölüm 1.4)
- **Daha tutarlı testler** ve daha yüksek seviyede hata tekrar oluşturulabilirliği sağlayarak test faaliyetlerinin **kalitesini iyileştirmek**
- Manuel olarak yürütülemeyen faaliyetleri otomatikleştirmek (örneğin, büyük ölçekli performans testleri)
- Testlerin güvenilirliğini artırmak (örneğin, büyük veri karşılaştırmalarını otomatikleştirerek veya davranışı simüle ederek)

Araçlar; amaç, fiyatlandırma, lisanslama modeli (örneğin, ticari veya açık kaynak) ve kullanılan teknoloji gibi çeşitli kriterlere göre sınıflandırılabilir. Bu ders programında **araçlar destekledikleri test faaliyetlerine göre sınıflandırılır.**

Bazı araçlar yalnızca veya temelde bir test faaliyetini desteklerken, bazıları birden fazla test faaliyetini destekleyebilir, ancak **en yakından ilişkili oldukları faaliyet altında sınıflandırılır**. Tek bir tedarikçinin sunduğu araçlar, özellikle birlikte çalışmak üzere tasarlanmış olanlar, entegre bir paket şeklinde sağlanabilir.
Bazı test araçları test sonucunu etkileyebilir cinste olabilir; bu, testin gerçekleşen çıktılarını etkileyebilecekleri anlamına gelmektedir. Örneğin,
bir uygulama için gerçekleşen yanıt süreleri, bir performans test aracı tarafından yürütülen ekstra komutlar nedeniyle farklılaşabilir veya bir kapsam aracı kullanılması nedeniyle ulaşılan kod kapsamı miktarı hatalı olabilir. Bu tür araçların kullanımı sebebiyle sonucun değişmesine gözlemci etkisi denir.

Bazı araçların, yazılımcılara olan desteği daha çoktur (örneğin, birim ve entegrasyon testleri sırasında kullanılan araçlar). Bu araçlar aşağıdaki bölümlerde "(D)" ile işaretlenmiştir.

Testlerin ve test yazılımının yönetimi için araç desteği

Yönetim araçları, tüm yazılım geliştirme yaşam döngüsü boyunca her test faaliyeti için kullanılabilir. Testlerin ve test yazılımının yönetimini destekleyen araçlara örnekler aşağıda verilmiştir:

- Test yönetim araçları ve uygulama yaşam döngüsü yönetimi araçları (ALM)
- Gereksinim yönetim araçları (örneğin, test nesneleri arasında izlenebilirliğin sağlanması için)
- Hata yönetim araçları(Defect management tools)
- Yapılandırma yönetimi araçları(Configuration management tools)
- Sürekli entegrasyon araçları(Continuous integration tools) (D)

(Testlerin ve test yazılımının yönetimini destekleyen araçlar; Test management tools and application lifecycle management tools, Requirements management tools, Defect management tools, Configuration management tools, Continuous integration tools)

Statik testler için araç desteği

Statik test araçları, 3. bölümde açıklanan faaliyetler ve faydalarla ilişkilidir. Bu araçlara örnekler aşağıda verilmiştir:

- Gözden geçirme destek araçları
- ***Statik analiz araçları*** (D)

Test tasarımı ve uyarlama için araç desteği

Test tasarım araçları, test tasarım ve uyarlamasında test senaryoları, test prosedürleri ve test verileri gibi çalışma ürünlerinin oluşturulmasına yardımcı olur. Bu araçlara örnekler aşağıda verilmiştir:

- Test tasarım araçları
- Model Bazlı test araçları
- Test verisi hazırlama araçları
- Kabul testi güdümlü yazılım geliştirme (ATDD) ve davranış güdümlü yazılım geliştirme (BDD) araçları
- Test güdümlü yazılım geliştirme (TDD) araçları (D)

(Test tasarımı ve uyarlama için araçlar; Test tasarım araçları, Model Bazlı test araçları, Test verisi hazırlama araçları, Kabul testi güdümlü(ATDD) ve davranış güdümlü(BDD) test araçları, Test güdümlü yazılım geliştirme(TDD))

Bazı durumlarda, test tasarımını ve uyarlamasını destekleyen araçlar aynı zamanda testin koşturulmasını ve kaydedilmesini de destekleyebilir veya test koşumunu ve kaydını destekleyen diğer araçlara doğrudan çıktılarını sağlayabilir.

Test koşumu ve kayıt için araç desteği

Test koşumu ve kaydetme faaliyetlerini desteklemek için birçok araç bulunmaktadır. Bu araçlara örnekler aşağıda verilmiştir:

- Test koşum araçları (ör. regresyon testleri koşturmak için)
- Kapsam araçları (ör. gereksinim kapsamı, kod kapsamı (D))
- Test kuluçkaları (D)
- Birim testi çerçeve araçları (D)

Performans ölçümü ve dinamik analiz için araç desteği

Performans ölçümü ve dinamik analiz araçları, performans ve yük testi faaliyetlerinin desteklenmesinde gereklidir, çünkü bu faaliyetler manuel olarak etkin bir şekilde yapılamaz. Bu araçlara örnekler aşağıda verilmiştir:

- Performans testi araçları
- İzleme araçları
- Dinamik analiz araçları (D)

Özel test ihtiyacı için araç desteği

Genel test sürecini destekleyen araçlara ek olarak, daha özel test faaliyetlerini destekleyen başka birçok araç da vardır. Bunlara örnek olarak aşağıdaki konulara odaklanan araçlar gösterilebilir:

- Veri kalitesini değerlendirme
- Veri dönüştürme ve taşıma
- Kullanılabilirlik testleri
- Erişilebilirlik testleri
- Yerelleştirme testleri
- Güvenlik testleri
- Taşınabilirlik testleri (örneğin, desteklenen birden fazla platformda yazılımın test edilmesi)

6.1.2 Test Otomasyonunun Faydaları ve Riskleri

Sadece test aracına odaklanıp bunu hayata geçirmeye çalışmak iyi bir test pratiğinin garantisi değildir. Bir kurumda kullanımına alınan her yeni araç, gerçek ve kalıcı faydalar sağlamak için çalışma gerektirmektedir. Testlerde araçların kullanılmasının potansiyel faydaları ve sunduğu fırsatların yanında bazı riskleri de vardır. Bu durum, özellikle test koşumu araçları (genellikle test otomasyonu olarak da adlandırılan) için geçerlidir.

Test koşumunu desteklemek için araç kullanmanın potansiyel faydaları aşağıda verilmiştir:

- Tekrarlanan **manuel testlerde azalma** (örneğin, regresyon testlerinin koşumu, ortam kurma/kaldırma işlerinin yapılması, aynı test verilerinin yeniden girilmesi ve kodlama standartlarına uygunluğun kontrol edilmesi), dolayısıyla zamandan tasarruf
- Daha fazla **tutarlılık ve tekrarlanabilirlik** (örneğin, test verileri tutarlı bir şekilde oluşturulur, testler bir araç tarafından, aynı sıklıkta aynı sırada yapılır ve testler tutarlı bir şekilde gereksinimlerden elde edilir)
- Daha **objektif değerlendirme** (örneğin statik ölçümler, kapsam)
- Testlerle ilgili **bilgilere daha kolay erişim** (örneğin, test ilerlemesi, hata oranları ve performansla ilgili istatistikler ve grafikler)

(Testleri execute etmek için tool kullanmanın faydaları; manuel testlerde azalma, zaman tasarrufu, tutarlılık ve tekrarlanabilirlik, objektif değerlendirme, bilgilere daha kolay erişim)

Testleri desteklemek için araç kullanmanın potansiyel riskleri aşağıda verilmiştir:

- Araç için **beklentiler gerçekçi olmayabilir** (sahip olduğu fonksiyonalite ve kullanım kolaylığı dâhil)
- Bir **aracın kullanılmaya başlanması için gerekecek zaman**, maliyet ve efor olması gerekenden daha az tahmin edilebilir (araçla ilgili alınacak eğitim ve dış uzman desteği dâhil)
- **Araçtan yüksek miktarda ve sürekli fayda elde etmek için gereken zaman ve efor olması gerekenden daha az olarak tahmin edilebilir** (test sürecinde değişiklik yapılması ve aracın kullanım şeklindeki sürekli iyileştirme ihtiyacı dâhil)
- Aracın ürettiği test varlıklarının **bakımı ve güncel tutulması için gereken efor olması gerekenden** daha az olarak tahmin edilebilir
- Araca **çok fazla bağlı olunabilir** (test tasarımının veya koşumunun yerini alacağı veya **manuel testlerin daha iyi olacağı yerlerde otomatik testlerin kullanılacağı düşünülebilir**).
- Test varlıklarının **versiyon kontrolü ihmal edilebilir**
- Gereksinim yönetimi araçları, yapılandırma yönetimi araçları, hata yönetimi araçları gibi kritik önemdeki araçlar ve birden çok tedarikçinin araçları arasındaki ilişkiler ve **birlikte çalışabilirlik sorunları gözden kaçırılabilir.**
- Aracın **tedarikçisi iflas edebilir, aracı desteklemeyi bırakabilir** veya aracı farklı bir kuruma satabilir
- **Tedarikçi**; bakım, destek, yeni versiyonlar ve hata düzeltmeleri için **yeterli desteği vermeyebilir**
- Açık kaynak kodlu bir yazılımın geliştirilmesi geçici olarak durdurulabilir

- Yeni bir platform veya teknoloji test aracı tarafından desteklenmeyebilir
- Aracın net bir sahibi olmayabilir (ör. güncellemeler için)

6.1.3 Test Koşumu ve Test Yönetim Araçlarına Dair Önemli Hususlar

Test aracının düzgün ve başarılı şekilde hayata geçirilmesi için, bir kurumda test koşumu ve test yönetimi araçlarını seçerken ve entegre ederken göz önünde bulundurulması gereken bazı hususlar vardır.

Test koşumu araçları(Test execution tools, Testausführungswerkzeuge)

Test koşumu araçları, otomatikleştirilmiş test betikleri(test scripts, Testscript: sistemin beklendiği gibi çalışıp çalışmadığını test etmek için test edilen sistemde gerçekleştirilecek bir dizi talimata denir) kullanarak test nesnelerini koşturur. Bu tür bir araç, genellikle belirgin bir fayda elde etmek için önemli ölçüde çalışma gerektirir.

Capturing test approach(Mitschnitt) Görüntü yakalama,kayıt test yaklaşımı: Manuel çalışan bir test uzmanının aksiyonlarının kaydedilerek testlerin kayıt altına alınması çekici görünebilir, ancak bu yaklaşım **yüksek miktarlarda test betiğini barındıran test senaryolarında verimli çalışmayabilir**. Kaydedilen her bir betik spesifik veriler ve aksiyonlarla doğrusal olarak çalışan bir kurgudur. Beklenmeyen bir olay gerçekleştiğinde bu tür betikler tutarlı davranmayabilir. "Akıllı" görüntü yakalama teknolojisinden yararlanan bu araçların en yeni nesli, bu araç sınıfının kullanışlılığını artırmıştır; ancak **test edilen yazılımın kullanıcı arayüzü zaman içinde değiştikçe ve geliştikçe, üretilen betikler, devamlı bakım gerektirmeye devam etmektedir.**
(yüksek miktarlarda test betiğini barındıran test senaryolarında verimli çalışmayabilir, test edilen yazılımın kullanıcı arayüzü zaman içinde değiştikçe ve geliştikçe, üretilen betikler, devamlı bakım gerektirmeye devam etmektedir)

Veri güdümlü test yaklaşımı(Data-driven test approach, Datengetrieben): test girdilerini ve beklenen sonuçları, genellikle bir elektronik tablo şeklinde ayırır ve giriş verilerini okuyabilen ve aynı test betiğini farklı verilerle çalıştırabilen daha genel bir test betiği kullanır. Betik dilini bilmeyen test uzmanları daha sonra bu önceden tanımlanmış betikler için yeni test verileri yaratabilir.

Aksiyon kelimesi güdümlü test yaklaşımında(Keyword-driven test approach, Schlüsselwortgetrieben): genel bir betik, gerçekleştirilecek işlemleri tanımlayan anahtar kelimeleri (bunlara aksiyon kelimeleri de denir) işler, ardından ilişkili test verilerini işlemek için anahtar kelime betiklerini çağırır. Test uzmanları (betik diline hakim olmasalar bile), anahtar kelimeleri ve ilgili verileri kullanarak testleri tanımlayabilirler, bunlar da test edilen uygulamaya göre uyarlanabilir. Veri güdümlü ve aksiyon kelimesi güdümlü test yaklaşımlarının ayrıntıları ve örnekleri, ISTQB-TAE İleri Seviye Test Otomasyon Mühendisi Ders Programı, Fewster 1999 ve Buwalda 2001'de verilmiştir.

Yukarıdaki yaklaşımlar, betik dilinde uzmanlaşmış birinin (test uzmanları, yazılımcılar veya test otomasyonu uzmanları) bulunmasını gerektirir.

Kullanılan betik oluşturma tekniğinden bağımsız olarak, her test için beklenen sonuçların testten elde edilen gerçekleşen sonuçlarla karşılaştırılması gerekir; bu dinamik olarak (test koşturulurken) gerçekleştirilebilir veya daha sonra (koşum sonrası) karşılaştırma için sonuçlar saklanır.

Model Bazlı test (MBT) araçları, bir fonksiyonel gereksinimin, aktivite diyagramı gibi bir model biçiminde kaydedilmesine olanak sağlar.
Bu görev genellikle bir sistem tasarımcısı tarafından gerçekleştirilir. MBT aracı, test senaryosu gereksinimlerini oluşturmak için modeli yorumlar; bu, daha sonra bir test yönetim aracına kaydedilebilir ve/veya bir test koşum aracı tarafından yürütülebilir (bkz. ISTQB-MBT Temel Seviye Model Bazlı Testler Ders Programı).

Test yönetim araçları(Test management tools, Testmanagementwerkzeuge)

Test yönetim araçlarının, aşağıdakiler de dâhil olmak üzere çeşitli nedenlerle, **sık sık diğer araçlarla veya elektronik tablolarla etkileşim halinde olması gerekir:**

- Kurumun ihtiyaçlarına uygun bir formatta faydalı bilgiler üretmek
- Gereksinim yönetim aracındaki gereksinimlerle izlenebilirliği devam ettirmek
- Yapılandırma yönetimi aracındaki test nesnesi versiyon bilgisiyle bağlantı kurmak

Bu, test yönetimi modülü (ve muhtemelen bir hata yönetimi sistemi) ve kurum içindeki farklı gruplar tarafından kullanılan diğer modüller (ör. proje zaman çizelgesi ve bütçe bilgileri) de dâhil olmak üzere entegre bir araç (ör. uygulama yaşam döngüsü yönetimi - ALM) kullanırken dikkate alınması gereken önemli bir konudur.
(Test manager modülü ve diğer entegre modüller sık sık diğer araçlarla veya elektronik tablolarla etkileşim halinde olması gerekir. Bu dikkate alınması gereken önemli bir konudur.)

6.2.1 Araç Seçiminde Ana Prensipler

Kurum için araç seçiminde dikkate alınması gereken ana hususlar şunlardır:

- Kurumun olgunluğunun, **güçlü ve zayıf yönlerinin değerlendirilmesi**
- Araçlar tarafından desteklenen gelişmiş bir test süreci için fırsatların belirlenmesi
- Bu teknolojiyle uyumlu bir araç seçmek için, test nesneleri tarafından kullanılan teknolojilerin anlaşılması
- Araç uyumluluğu ve entegrasyonunu sağlamak için **hâlihazırda** organizasyon içinde **kullanımda olan derleme ve sürekli entegrasyon araçları**
- **Aracın** açıkça belirtilmiş **gereksinimlere ve objektif kriterlere** göre **değerlendirilmesi**
- Aracın **ücretsiz bir deneme süresi** boyunca (ve ne kadar süreyle) kullanılabilir olup olmadığı konusu
- **Tedarikçi değerlendirmesi** (eğitim, destek ve ticari yönler dâhil) veya ticari olmayan (ör. açık kaynak) araçlar için destek
- Aracın **kullanımında koçluk, mentorluk ve destek** için dâhili gereksinimlerin belirlenmesi
- Doğrudan araçla/araçlarla çalışacak kişilerin test (ve test otomasyonu) becerilerini dikkate alarak **eğitim ihtiyaçlarının** değerlendirilmesi

- Çeşitli **lisanslama modellerinin** (örneğin, ticari veya açık kaynak) artıları ve eksilerinin **değerlendirilmesi**
- Somut bir iş senaryosuna dayanan bir **fayda/maliyet oranının tahmini** (gerekirse)

(Kurum için araç seçiminde dikkate alınması gereken; Kurumun güçlü, zayıf yönleri. Uyumlu araç seçebilmek için seçilecek teknolojinin anlaşılması. Mevcut kullanılan toollar ile entegrasyon için derleme ve sürekli entegrasyon araçları olacak mı. Arac kurum gereksinimlerine ve kurumun objektif kriterlerine uyuyor mu. Ücretsiz deneme süresi var mı. Tedarikçi değerlendirmesi. Koçluk, mentörlük desteği olacak mı. Personelin eğitim ihtiyaçları karşılanacak mı. Lisanslama modellerinin değerlendirmesi. Kuruma fayda/maliyet katkısı ne olur)

Son adım olarak, aracın test edilen yazılımla ve mevcut altyapı dâhilinde etkin bir şekilde çalışıp çalışmadığını belirlemek veya gerekiyorsa aracı etkin şekilde kullanmak amacıyla bu altyapıda gereken değişiklikleri belirlemek için kavram kanıtlama (POC) değerlendirmesi yapılmalıdır.

(Kurumda yeni bir test aracı kullanılması düşünülüyorsa mevcut altyapı dahilinde bu yeni tool'un etkin bir şekilde çalışıp çalışmayacağını belirlemek veya gerekiyorsa aracı etkin şekilde kullanmak amacıyla mevcut altyapıda değişiklikler yapmak gerekecetir. Yapılması gereken bu değişikleri belirlemek için yapılan değerlendirmeye* kavram kanıtlama (POC) değerlendirmesi(proof-of-concept evaluation) *denir)

6.2.2 Bir Kurumda Bir Aracı Uygulamaya Koymak için Pilot Projeler

Araç seçimi ve başarılı bir kavram kanıtlama(POC) tamamlandıktan sonra, seçilen aracın kurumda kullanıma başlanması genellikle pilot bir projeyle başlar, bu proje ile aşağıdakiler hedeflenir:

- Araç hakkında derinlemesine bilgi edinmek, güçlü ve zayıf yönlerini anlamak.
- Aracın mevcut süreç ve uygulamalara nasıl uyacağını değerlendirmek ve gereken değişiklikleri belirlemek.
- Aracı ve test varlıklarını kullanmanın, yönetmenin, saklamanın ve bakımını yapmanın yol ve yöntemlerine karar vermek. (örneğin, dosyalar ve testler için adlandırma kurallarına karar vermek, kodlama standartlarını seçmek, kütüphaneler oluşturmak ve test gruplarının modülerliğini belirlemek)
- Aracın sağladığı faydaların kabul edilebilir bir maliyet ile sağlanıp sağlanamayacağını değerlendirmek.
- Aracın toplamasını ve raporlamasını istediğiniz metrikleri anlamak ve bu metriklerin kaydedilip raporlanabilmesini sağlamak için aracı yapılandırmak.

6.2.3 Araçlar için Başarı Faktörleri

Bir kurum içindeki araçların değerlendirilmesi, uyarlanması, dağıtımı ve devamlı desteği için başarı faktörleri aşağıdakileri içerir:

- Aracın kurumun geri kalanına aşamalı olarak yayılması
- Aracın kullanımı için süreçleri uyarlamak ve geliştirmek
- Araç kullanıcıları için eğitim, koçluk ve mentorluk sağlamak
- Aracın kullanımı için yönergeler tanımlamak (örneğin, otomasyon için dâhili standartlar)
- Aracın kullanımı süresince elde edilen bilgileri toplamak için bir yöntem belirlemek
- Araç kullanımını ve faydalarını izlemek
- Belirli bir aracın kullanıcılarına destek sağlamak
- Kullanıcılardan geribildirimleri toplamak

Ayrıca, aracın yazılım geliştirme yaşam döngüsüne teknik ve organizasyonel olarak entegre olmasını sağlamak önemlidir; bunun için yönetimden ve/veya tedarikçilerden bu işle sorumlu ayrı kurumların katılımı gerekebilir.

Test koşumu araçlarını kullanmakla ilgili deneyimler ve tavsiyeler için Graham 2012'ye bakınız.