Sertifikalı Test Uzmanı Temel Seviye Ders Programı

Yayın
Versiyon 2011

International Software Testing Qualifications Board

Telif Hakkı Bildirimi
Kaynağı gösterildiği takdirde dokümanın tümü veya bir kısmı kullanılabilir veya kopyalanabilir

Telif Hakkı Bildirimi © International Software Testing Qualifications Board (bundan sonra ISTQB® olarak anılacaktır) ISTQB, International Software Testing Qualifications Board firmasının tescilli ticari markasıdır,

Telif Hakkı © 2011, 2011 güncellemesinin yazarları (Thomas Müller (başkan), Debra Friedenberg ve ISTQB Çalışma Grubu Temel Seviye)

Telif Hakkı © 2010, 2010 güncellemesinin yazarları (Thomas Müller (başkan), Armin Beer, Martin Klonk, Rahul Verma)

Telif Hakkı © 2007, 2007 güncellemesinin yazarları (Thomas Müller (başkan), Dorothy Graham, Debra Friedenberg ve Erik van Veenendaal)

Telif Hakkı © 2005, yazarlar (Thomas Müller (başkan), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson ve Erik van Veenendaal.

Tüm hakları saklıdır.

İşbu yazarlar telif hakkını International Software Testing Qualifications Board'a (ISTQB) devretmektedir. Yazarlar (geçerli telif hakkı sahibi olarak) ve ISTQB (gelecekteki telif hakkı sahibi olarak), aşağıdaki kullanım koşullarını kabul etmiştir:

1) Bu ders programı, yazarlar ve ISTQB kaynak ve telif hakkı sahipleri olarak gösterildiği sürece herhangi bir kişi veya eğitim kurumu tarafından eğitim kursunun temeli olarak kullanılabilir. Ders programının kullanılmasının diğer bir şartı ise söz konusu eğitim programıyla ilgili yapılacak herhangi bir reklamda bu ders programından bahsedilmeden önce bu ders programı baz alınarak geliştirilmiş eğitim materyallerinin resmi akreditasyon için ISTQB tarafından tanınmış bir Ulusal Kurula iletilmiş olmasıdır.
2) Yazarlar ve ISTQB bu ders programının kaynağı ve telif hakkı sahibi olarak gösterildiği sürece, herhangi bir kişi veya bir grup bu ders programını makale, kitap veya diğer yazılar için temel olarak kullanabilir.
3) ISTQB tarafından tanınan herhangi bir Ulusal Kurul bu ders programını tercüme edebilir ve ders programının (ya da tercümesinin) lisansını başkalarına verebilir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Revizyon Geçmişi

Versiyon	Tarih	Notlar
ISTQB 2011	1 Nisan 2011 itibarıyla geçerli	Sertifikalı Test Uzmanı Temel Seviye Ders Programı Bakım Sürümü – bkz. Ek E – Sürüm Notları
ISTQB 2010	30 Mart 2010 itibarıyla geçerli	Sertifikalı Test Uzmanı Temel Seviye Ders Programı Bakım Sürümü – bkz. Ek E – Sürüm Notları
ISTQB 2007	1 Mayıs 2007	Sertifikalı Test Uzmanı Temel Seviye Ders Programı Bakım Sürümü
ISTQB 2005	1 Temmuz 2005	Sertifikalı Test Uzmanı Temel Seviye Ders Programı
ASQF V2.2	Temmuz 2003	ASQF Ders Programı Temel Seviye Versiyon 2.2 “Lehrplan Grundlagen des Software-testens“
ISEB V2.0	25 Şubat 1999	ISEB Yazılım Testi Temel Ders Programı V2.0 25 Şubat 1999

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Önsöz

Günümüz iş hayatında ünvanları test uzmanı, test mühendisi, test takım lideri ve test yöneticisi olan binlerce profesyonel olmasına rağmen ülkemizde test ve test kavramlarıyla ilgili Türkçe kaynak malesef yeteri seviyede bulunmamaktadır. Bu açığı kapatmak üzere Yazılım Test ve Kalite Derneği (www.turkishtestingboard.org) kendi bünyesinde bir çalışma başlatarak kar amacı gütmeyen ve dünyanın en saygın yazılım test organizasyonlarından olan ISTQB'nin (International Software Testing Qualifications Board – www.istqb.org) Certified Tester Foundation Level Syllabus Version 2011'i Türkçeleştirmiştir. Türkçeleştirme çalışması yapılırken daha önceden çevirisi yapılmış ISTQB Yazılım Testi Terimler Sözlüğü baz alınmış, çevirinin yazılım test sektöründe kullanılan Türkçe'ye uygun olmasına dikkat edilmiştir. Bu özene rağmen dilin yaşayan bir varlık olduğu, sürekli değiştiği, İngilizcedeki bazı kelimelerin Türkçemizde tam karşılığının olmadığı ve yazılım test sektöründe terimlerin halen standartlaşmadığı gözardı edilmemelidir. Bu kısıtlardan dolayı çevirinin yaşanan gelişmeler ışığında her zaman güncel olabilmesi için yeni gelişmeleri ve önerilerinizi info@turkishtestingboard.org e-posta adresine gönderebilirsiniz.

Çevirinin Türkiye bilişim sektörüne faydalı olması dileğiyle.

Yazılım Test ve Kalite Derneği
Eylül, 2014

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Teşekkür

ISTQB Sertifikalı Test Uzmanı Temel Seviye Ders Programının Türkçeleştirme çalışmasına katkıda bulunan Yazılım Test ve Kalite Derneği Temel Seviye Ders Programı çalışma grubu üyelerine burada tekrar teşekkür etmek isteriz. Çalışma grubu üyeleri (alfabetik sıraya göre):

- Adem Çağlar
- Ahmet Zeki Boyraz
- Ali Dağdemir
- Alper Mermer
- Aydın Akkaya
- Barış Sarıalioğlu
- Barış Hakkı Tokmak
- Bora Sipal
- Deniz Tezer
- Emrah Yayıcı
- Engin Başarslan
- Fatma Molu
- Gizem Gürcüoğlu
- Halil Yaman Manargalı
- Hasan Ulusoy
- Kadir Herkiloğlu
- Kemal Ergezer
- Koray Yitmen
- Mehmet Nuri Gülöksüz
- Merve İçöz
- Onur Güngör
- Onur Sertel
- Ömer Hamdi Kaya
- Savaş Oltekin
- Selin Hekimoğlu
- Sera Seren
- Turgay Aksaray
- Ünzile Algül
- Volkan Arısoy

İçindekiler

Teşekkür

International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Baskı 2011): Thomas Müller (başkan), Debra Friedenberg. Temel seviye çalışma grubu ders programıyla ilgili önerileri için gözden geçirme ekibine (Dan Almog, Armin Beer, Rex Black, Julie Gardiner, Judy McKay, Tuula Pääkkönen, Eric Riou du Cosquier Hans Schaefer, Stephanie Ulrich, Erik van Veenendaal) ve tüm ulusal kurullara teşekkürlerini sunar.

International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Baskı 2010): Thomas Müller (başkan), Rahul Verma, Martin Klonk ve Armin Beer. Temel seviye çalışma grubu önerileri için gözden geçirme ekibine (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Judy McKay, Tuula Pääkkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veenendaal) ve tüm ulusal kurullara teşekkürlerini sunar.

International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Baskı 2007): Thomas Müller (başkan), Dorothy Graham, Debra Friedenberg, ve Erik van Veenendaal. Temel seviye çalışma grubu , önerileri için gözden geçirme ekibine (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson ve Wonil Kwon) ve tüm ulusal kurullara teşekkürlerini sunar.

International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Baskı 2005): Thomas Müller (başkan), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson ve Erik van Veenendaal ve gözden geçirme ekibi ile tüm ulusal kurullara önerileri için teşekkürlerini sunar.

Ders Programına Giriş

Amaç

Bu ders programının amacı temel seviye uluslararası yazılım test uzmanı niteliklerine yönelik bir çerçeve oluşturmaktır. International Software Testing Qualifications Board (ISTQB), eğitim sağlayıcıları akredite etmeleri ve yerel dillerinde sınav soruları türetmeleri için bu ders programını ulusal kurullara sunmaktadır. Eğitim sağlayıcılar uygun öğretme yöntemlerini belirledikten sonra bu ders programını kullanarak akreditasyon için eğitim malzemeleri hazırlayabilirler. Ders programı, aynı zamanda sertifikalı test uzmanı adaylarının sınava hazırlanmasına yardımcı da olmaktadır. Ders programının geçmişi ve genel bilgiler Ek A'da bulunabilir.

Yazılım Testinde Sertifikalı Test Uzmanı - Temel Seviye

Temel seviye ders programı yazılım testine dahil veya yazılım testi ile uğraşmış olan herkese yöneliktir. Bunlar arasında, test uzmanları, test analistleri, test mühendisleri, test danışmanları, test yöneticileri, kullanıcı kabul testi uzmanları ve yazılımcılar bulunur. Temel seviye nitelik olarak aynı zamanda, yazılım testinin temel kavramlarını anlamak isteyen ve yazılım geliştirme projelerine dahil olan proje yöneticileri, kalite güvence uzmanları, yazılım geliştirme yöneticileri, iş analistleri, sistem analistleri, kullanılabilirlik uzmanları, BT direktörleri ve yönetim danışmanları için de uygundur. Bu ders programına ilişkin sınava girerek temel seviye sertifikaya sahip olan kişiler, ileri seviye test uzmanı ders programlarına devam edebilirler.

Öğrenme Hedefleri/Bilişsel Bilgi Seviyesi

Öğrenme hedefleri her bir kısım için bu ders programında aşağıdaki şekilde sınıflandırılmıştır:

- K1: hatırlanması gereken
- K2: anlamanın yeterli olduğu
- K3: uygulama gerektiren
- K4: analiz edilmesi gereken

Öğrenme hedefleri ile ilgili daha fazla ayrıntı ve örnekler Ek B'de verilmektedir.

Bölüm başlıklarının altındaki "Terimler" başlığı altında listelenen tüm terimler, öğrenme hedeflerinde açıkça belirtilmese dahi hatırlanmalıdır. (K1).

Sınav

Temel seviye sertifika sınavı, bu ders programını baz alacaktır. Sınav sorularına verilecek yanıtlar, bu ders programının birden fazla kısmını ilgilendirebilir. Ders programının tüm bölümlerinden sınav yapılabilir.

Sınav çoktan seçmelidir.

Test uzmanı adayları sınavlara, akredite bir eğitimin bir parçası olarak veya bağımsız bir şekilde (örneğin bir sınav merkezinde veya genel bir sınav şeklinde) girebilirler. Akredite bir eğitimin tamamlanması, sınav için ön şart değildir.

Akreditasyon

Bir ISTQB ulusal kurulu, ders materyalleri bu ders programına uyumlu eğitim sağlayıcıları akredite edebilir. Eğitim sağlayıcılar, kuruldan veya akreditasyon veren kuruluştan akreditasyon kurallarını almalıdır. Akredite bir eğitimin bu ders programına uygun olduğu kabul edilir ve eğitimin bir parçası olarak ISTQB sınavının gerçekleştirilmesine izin verilir.

Eğitim sağlayıcılar ile ilgili daha fazla bilgi Ek D'de verilmektedir.

Ayrıntı Seviyesi

Bu ders programındaki ayrıntı seviyesi, uluslararası tutarlılıkta öğretim yapılmasına ve sınav sorusu oluşturulmasına imkan verir. Bu hedefe ulaşmak için ders programı şunları içerir:

- Temel seviyenin amacını açıklayan genel eğitim hedefleri
- Öğretilecek konular hakkında bir tanım da içeren bilgi listesi ve gerekiyorsa, kullanılması gereken ek kaynaklara referanslar
- Elde edilecek bilişsel öğrenme ürününü/çıktısını ve anlayışını tanımlayan, her bir bilgi alanına yönelik öğrenme hedefleri
- Öğrencilerin hatırlaması ve anlaması gereken terimler listesi
- Literatürde genel geçer kabul görmüş kaynakların ve standartların da dahil olduğu anahtar kavramların açıklamaları

Ders programı, yazılım testi alanı ile ilgili tüm bilgilerin bir açıklaması değildir, bu program ISTQB temel seviye eğitimlerinde kapsanacak ayrıntı seviyesini yansıtır.

Ders Programının Düzeni

Altı ana bölüm bulunmaktadır. Her bir bölümün üst başlığı, bölümde açıklanan öğrenme hedeflerinin en üst seviyesini gösterir ve bölümün süresini belirtir. Örneğin:

2. Yazılım Yaşam Döngüsü Boyunca Test (K2) 115 dakika

Bu başlık, Bölüm 2'de K1 (daha yüksek bir seviye olan K2 belirtildiği için) ve K2 seviyesinde öğrenme hedeflerinin bulunduğunu ve bölümdeki materyalleri öğretmenin 115 dakika civarında süreceğini gösterir. Her bir bölümün içinde birçok kısım bulunur. Ayrıca her kısımda da öğrenme hedefleri için gereken süre belirtilir. Süre verilmeyen alt kısımlar, o bölüm için belirlenen toplam süreye dahildir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

1. Test ile İlgili Temel Bilgiler (K2) — 155 dakika

Test ile İlgili Temel Bilgiler için Öğrenme Hedefleri

Aşağıda listelenen hedefler, her bir modülü tamamladıktan sonra test uzmanı adayının neler yapabileceğini tanımlar.

1.1 Test Neden Gerekli? (K2)

ÖH-1.1.1 Yazılımdaki bir hatanın, kişiye, çevreye veya şirkete ne şekilde zarar verebileceğini örneklerle açıklama (K2)

ÖH-1.1.2 Bir hatanın kök nedeni ile etkileri arasındaki farkı ayırt etme (K2)

ÖH-1.1.3 Örnekler vererek testin neden gerekli olduğunu açıklama (K2)

ÖH-1.1.4 Testin neden kalite güvencenin bir parçası olduğunu açıklama ve test etmenin daha yüksek kaliteli bir yazılım elde etmeye nasıl katkı sağladığı ile ilgili örnekler verme (K2)

ÖH-1.1.5 İnsan hatası, hata, kusur, arıza ve yanlış ile yazılım hatası terimlerini örneklerle açıklama ve karşılaştırma (K2)

1.2 Test Nedir? (K2)

ÖH-1.2.1 Testin ortak hedeflerini hatırlama (K1)

ÖH-1.2.2 Yazılım yaşam döngüsünün farklı aşamalarında test hedefleri ile ilgili örnekler verme (K2)

ÖH-1.2.3 Test ile hata ayıklama arasındaki farkı belirtme (K2)

1.3 Yedi Test İlkesi (K2)

ÖH-1.3.1 Testin yedi ilkesini açıklama (K2)

1.4 Temel Test Süreci (K1)

ÖH-1.4.1 Planlamadan kapanışa kadar beş temel test sürecini ve ilgili adımları hatırlama (K1)

1.5 Test Psikolojisi (K2)

ÖH-1.5.1 Testin başarısını etkileyen psikolojik faktörleri hatırlama (K1) Bir test uzmanı ile yazılımcının

ÖH-1.5.2 bakış açılarını karşılaştırma (K2)

1.1 Test Neden Gerekli? (K2)	*20 dakika*

Terimler

Yazılım hatası, hata, insan hatası, arıza, kusur, yanlış, kalite, risk

1.1.1 Yazılım Sistemleri (K1)

Yazılım sistemleri, kurum içi uygulamalardan (örn. bankacılık) tüketici ürünlerine kadar (örn. otomobiller), yaşamın ayrılmaz bir parçasıdır. Birçok insan beklendiği gibi çalışmayan bir yazılımla karşılaşmıştır. Doğru şekilde çalışmayan yazılım, para, zaman ve itibar kaybı gibi birçok probleme yol açabilir ve hatta yaralanmaya ve ölüme neden olabilir.

1.1.2 Yazılım Hatalarının Nedenleri (K2)

İnsan hata (yanlış) yapabilir, bu hata da program kodunda veya bir dokümanda hatanın (kusur, yazılım hatası) oluşmasına sebep olur. Kodda bir hata olması durumunda sistem yapması gereken işlemi yerine getirmede başarısız olur (veya yapmaması gereken şeyi yapar) ve bu da arızaya neden olur. Yazılımdaki, sistemlerdeki veya dokümanlardaki hatalar, arızalara neden olabilir, ancak tüm hatalar arızaya sebep olacak diye bir kural yoktur.

Hataların oluşma sebebi insanların yanılabilmesi, zaman baskısı, karmaşık kodlar, altyapı karmaşıklığı, değişen teknolojiler ve/veya birçok sistemin etkileşimi olabilir.

Arızaların sebebi çevresel koşullar da olabilir. Örneğin, radyasyon, manyetizma, elektronik alanlar ve kirlilik, bu etkenler yazılımda kusurlara neden olabilir veya donanım koşullarını değiştirerek yazılımın yürütülmesini etkileyebilir.

1.1.3 Yazılım Geliştirme, Bakım ve Operasyonlarda Testin Rolü (K2)

Yazılımlar dikkatli şekilde test edildiği takdirde, hataları bulunup canlıya alınmadan önce düzeltmeler yapılabilir. Bu sayede arıza oluşma riski azalır ve yazılımın kalitesi artar.

Aynı zamanda yazılımların sözleşme, yasal gereksinimleri ya da endüstriye özel standartları karşılamak için de test edilmesi gerekebilir.

1.1.4 Test ve Kalite (K2)

Test sayesinde yazılımın hem fonksiyonel hem de fonksiyonel olmayan gereksinimleri (örn. güvenilirlik, kullanılabilirlik, verimlilik, sürdürülebilirlik ve taşınabilirlik) açısından kalite seviyesini belirlemek de mümkündür. Fonksiyonel olmayan test hakkında daha fazla bilgi için bkz. Bölüm 2, yazılım karakteristikleri ile ilgili daha fazla bilgi için bkz. "Yazılım Mühendisliği – Yazılım Ürün Kalitesi" (ISO 9126).

Testte bulunan hataların sayısı, bu hataların önemi ve önceliği yazılımın kalitesi hakkında güven sağlamaya yardımcı olur. Doğru şekilde tasarlanmış olan ve başarıyla geçen bir test, bir yazılımdaki genel risk algısının düşmesini sağlar. Test sonucunda bulunan hatalar düzeltildiği takdirde yazılımın kalitesi artar.

Yapılan test projelerinden dersler çıkarılmalı, bir sonraki test projelerine bu tecrübeler aktarılmalıdır. Diğer projelerde bulunan hataların kök nedenleri incelenerek süreçler iyileştirilebilir ve böylece bu hataların tekrar oluşması önlenir. Sonuç olarak ileride geliştirilecek yazılımların kalitesinde iyileşme sağlanmış olur. Bu süreç kalite güvencenin bir parçasıdır.

Test, yazılım geliştirme sürecine kalite güvence aktivitelerinin bir parçası olarak entegre edilmelidir. (örn. geliştirme standartları, eğitim ve hata analizi gibi)

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

1.1.5 Yazılımı Ne Kadar Test Etmek Gerekir? (K2)

Ne kadar testin yeterli olacağına karar vermek için riskleri, zaman ve bütçe gibi proje kısıtlarını göz önünde bulundurmak gerekir. Risk konusu Bölüm 5'te daha ayrıntılı şekilde incelenmektedir.

Test sonuçları, paydaşlara yazılımın piyasaya sürülme kararı, bir sonraki yazılım geliştirme aktivitesine geçiş veya müşteriye devri gibi karar süreçlerinde yardımcı olmalı, yeterince bilgi sağlamalıdır.

1.2 Test Nedir? (K2)	*30 dakika*

Terimler

Hata ayıklama, gereksinim, gözden geçirme, test senaryosu, test etme, test hedefi

Temel Bilgi

Test ilgili yaygın bir anlayış testin sadece test koşumu işleminden ibaret olduğu yönündedir. Test koşumu, test etme işleminin bir parçasıdır fakat tümü değildir.

Test sürecinde test koşumu öncesi ve sonrasında gerçekleştirilen başka işlemler de vardır. Bu işlemler arasında planlama ve kontrol, test koşullarını seçme, test senaryolarını tasarlama ve koşturma, sonuçları kontrol etme, çıkış kriterini değerlendirme, test süreci ve test edilen sistem ile ilgili raporlama ve test fazı tamamlandıktan sonra kapanış işlemlerini sonlandırma ve tamamlama yer alır. Test ayrıca, dokümanları gözden geçirme (kaynak kod dahil) ve statik analiz işlemlerini de içerir.

Dinamik test ve statik test benzer hedeflere ulaşmak için kullanılabilir ve test edilen sistem ile yazılım geliştirme ve test süreçlerini iyileştirmek için kullanılabilecek bilgiler sağlar.

Testin hedefleri aşağıdaki gibi olabilir:

- Hataları bulma
- Kalite seviyesi hakkında güven kazanma
- Karar verme için bilgi sağlama
- Hataları önleme

Kalite kontrol edilmez üretilir düşünce yapısı ve testlerin yazılım geliştirme sürecinin en başından itibaren işin içine dahil olması gerekliliği hataların daha yapılmadan önlenmesini sağlayacaktır. Üretilen çıktıların gözden geçirilmesi sonucunda (örn. analiz dokümanı) bulunan hataların tanımlanıp çözülmesi de kodda hataların çıkmasını engellemeye de yardımcı olabilir.

Test seviyelerindeki farklı bakış açıları farklı test amaçlarını ortaya çıkarabilir. Örneğin entegrasyon testinde amaç mümkün olan en fazla sayıda arızayı ortaya çıkarmak olabilir. Kullanıcı kabul testinde ise amaç yazılımın gereksinimleri karşılaması konusunda güven elde etmek olabilir. Bazı test senaryolarında ise testin amacı yazılımın kalitesini değerlendirmek (hataları düzeltme amacı olmadan) ve yazılımın belirli bir sürede piyasaya sürülmesi ile ilgili riskler konusunda paydaşlara bilgi vermek olabilir.

Bakım testi ise genellikle yazılımda yapılan iyileştirme ve değişikliklerin yeni bir hataya yol açıp açmadığını ortaya çıkarmak için yapılır. Operasyonel test sırasında ise amaç güvenilirlik veya elverişlilik gibi yazılım özelliklerini denetlemek olabilir.

Hata ayıklama ve test farklı kavramlardır. Dinamik test, hataların neden olduğu arızaları gösterebilir. Hata ayıklama ise arızanın sebebini bulan, analiz eden ve ortadan kaldıran bir çeşit yazılım geliştirme aktivitesidir. Düzeltme yapıldıktan sonra test uzmanı tarafından gerçekleştirilen tekrar testi, düzeltmenin arızayı giderip gidermediğinden emin olmak için yapılır. Bu aktivitede genellikle test uzmanları testi yapar, yazılımcılar ise hata ayıklar.

Test süreci ve test aktiviteleri, Kısım 1.4'te açıklanmaktadır.

1.3 Yedi Test Prensibi (K2)	*35 dakika*

Terimler

Yazılımı %100 test etmek

Prensipler

Geçen 40 yılda birçok test prensibi ortaya atılmış ve tartışılmıştır. Aşağıda listelenen prensipler bu prensiplerin ortak noktalarını biraraya getiren ve özetleyen bir özelliğe sahiptir

1. Testin amacı, yazılımda hataların olduğunu göstermektir; yazılımda hata kalmadığını ispatlamak değildir

Test aktiviteleri ile hatalar tespit edilebilir fakat testler yazılımda hata kalmadığını ispatlamak için yapılmaz. Test yazılımdaki hata sayısını düşürmesine rağmen; bu durum yazılımın, müşterilerin/kullanıcıların ihtiyaçlarını karşılayacağı anlamına gelmez.

2. Yazılımı %100 test etmek imkansızdır

Yazılımı tüm girdi ve çıktı kombinasyonları ile test etmek, projenin zaman ve bütçe kısıtları sebebiyle mümkün değildir. Önemli olan risk analizi ile yazılımdaki riskli alanların tespit edilip, önceliklendirmelerin yapılması ve bu alanların normalden daha fazla test edilmesidir.

3. Teste yazılım geliştirme sürecinin başında başlamak gerekir

Yazılım hatalarının erken fazlarda tespit edilmesi, bu hataların çözüme kavuşturulma maliyetlerini ciddi oranda azaltması sebebiyle kritiktir. Bu sebeple test aktivitelerine yazılım projelerinin erken fazlarında (örn. planlama, analiz, tasarım) başlanılması ve test ekiplerinin proje süreçlerine erken dahil edilmesi oldukça önemlidir.

4. Hatalar yazılımın belli alanlarında yoğunlaşır

Yazılımda hatalar belli alanlarda yoğunlaşmıştır. Yazılımın tüm parçaları (modülleri) dikkate alınacak olunursa, hataların büyük çoğunluğunun bu parçaların küçük bir kısmında bulunacağı görülecektir.

5. Antibiyotik Direnci

Test senaryolarının belirli aralıklarla güncellenmesi ve revize edilmesi gerekmektedir. Bunun yanında yazılımın farklı ve daha önce test edilmemiş parçalarını test eden yeni test senaryolarının hazırlanması, yeni test tekniklerinin kullanılması yazılımda daha fazla hatanın tespitine olanak sağlayacaktır.

6. Test yaklaşımı ve aktiviteleri yazılım projesinin koşullarına göre değişiklik gösterir

Test aktiviteleri, yazılımın özelliklerine, bağlamına ve içeriğine göre farklı biçimlerde ele alınmalıdır. Örnek olarak, bir e-ticaret yazılımı ile nükleer santral için yazılmış güvenlik tehlikesi taşıyan bir uygulama farklı şekillerde, farklı test teknikleri ve metodolojileri kullanılarak test edilmelidir.

7. Yeni hata bulamıyoruz başarılı bir yazılım elde ettik yanılgısı

Testte tespit edilen hataların düzeltilmiş olması, yazılımın müşteri/kullanıcı ihtiyaçlarını tam, eksiksiz ve doğru karşılıyor olduğu anlamını taşımamaktadır.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

1.4 Temel Test Süreci (K1)	*35 dakika*

Terimler

Onaylama testi, tekrar testi, çıkış kriteri, olay, regresyon, test esası, test koşulu, test kapsamı, test verisi, test yürütme, test kaydı, test planı, test prosedürü, test politikası, test grubu, test özet raporu, test yazılımı

Temel Bilgi

Testin en görünür bölümü test yürütmedir. Fakat verimli ve etkin bir test süreci işletebilmek için testin planlanması, test tasarım tekniklerinin kullanılarak test senaryolarının oluşturulması, test ortamının ve test verilerinin hazırlanması, test sonuçlarının değerlendirilip raporlanması ve kazanılan tecrübelerin bir sonraki projelere aktarılması gerekmektedir.

Temel test süreci aşağıdaki ana aktivitelerden oluşur:

- Test planlama ve kontrol
- Test analizi ve tasarımı
- Test uyarlama ve yürütme
- Çıkış kriterlerini değerlendirme ve raporlama
- Test kapanışı işlemleri

Süreçteki işlemler mantıksal sıradadır, ancak birbirleriyle kesişebilir veya birbirinin yerini alabilir. Genellikle bu temel aktivitelerin sistem ve proje bağlamına göre uyarlanması gerekir.

1.4.1 Test Planlama ve Kontrol (K1)

Test planlama, hedefleri ve misyonu karşılamak amacıyla testin amacını ve test aktivitelerinin detaylarını belirleme aktivitesidir.

Test kontrol, elde edilen ilerlemeyi planla karşılaştırma ve planlardan sapmalar da dahil olmak üzere durumu raporlama aktivitesidir. Projenin misyonunu ve hedeflerini karşılamak için gerçekleştirilmesi gereken aktiviteleri içerir. Testin kontrol edilmesi için test aktivitelerinin proje boyunca gözlemlenmesi gerekir. Test planlama, gözlemleme ve kontrol işlemlerinden gelen geri bildirimleri göz önüne alır.

Test planlama ve kontrol sürecinin detayları bu ders programı kapsamındaki Bölüm 5'te tanımlanmaktadır.

1.4.2 Test Analizi ve Tasarımı (K1)

Test analizi ve tasarımı, genel test hedeflerinin somut test koşullarına ve test senaryolarına dönüştürüldüğü işlemdir.

Test analizi ve tasarımı aşamasında aşağıdaki temel adımlar yer almaktadır:

- Test esasını gözden geçirme (gereksinimler, yazılımın bütünlük seviyesi[1] (risk seviyesi), risk analizi raporları, mimari, tasarım, arayüz özellikleri gibi)
- Test esasının ve test nesnelerinin test edilebilirliğini değerlendirme
- Test öğelerinin analizine, özelliklerine, davranışlarına ve yazılımın yapısına dayanan test koşullarını tanımlama ve önceliklendirme
- Üst seviye test senaryoları tasarlama ve önceliklendirme
- Test koşullarını ve test senaryolarını desteklemek için gerekli test verisini belirleme
- Test ortamının kurulumunu tasarlama ve gerekli altyapı ve araçları tanımlama
- Test esası ve test senaryoları arasında çift yönlü izlenebilirlik oluşturma

[1] Bir yazılımın önemini paydaşlarına yansıtmak için tanımlanmış olan yazılım ve / veya yazılım tabanlı bir sistemin bir dizi özellikleri ile uyumlu olduğu ya da olması gerektiği derece (örneğin, yazılım karmaşıklığı, risk değerlendirmesi, emniyet seviyesi, güvenlik seviyesi, istenilen performans, güvenilirlik veya maliyet).

1.4.3 Test Uyarlama ve Yürütme (K1)

- Test uyarlama ve yürütme, test senaryoları ve test verileri gibi diğer gerekli tüm bilgilerin birleştirilerek test prosedürlerinin veya test komut dosyalarının belirlendiği, test ortamının kurulduğu ve testlerin çalıştırıldığı aşamadır.
- Test uyarlama ve yürütme işleminde aşağıdaki temel adımlar yer almaktadır:
- Test senaryolarını yazma, uyarlama ve önceliklendirme (test verisinin tanımlanması da dahil)
- Test prosedürlerini yazma ve önceliklendirme, test verisi oluşturma ve isteğe bağlı olarak test kuluçkası hazırlama ve otomasyon için test komut dosyası yazma
- Verimli bir test yürütme işlemi için test prosedürlerinden test grupları oluşturma
- Test ortamının doğru şekilde kurulduğunu doğrulama
- Test esası ve test senaryoları arasındaki iki yönlü izlenebilirliği doğrulama ve güncelleme
- Planlanan sıraya göre test prosedürlerini manuel olarak veya test otomasyon araçları kullanarak yürütme
- Test otomasyon araçlarının çıktısını alma ve test edilen yazılımın özelliklerini ve versiyonlarını tutma, kullanılan test araçlarını ve test yazılımının kaydını tutma
- Gerçekleşen sonuçları beklenen sonuçlarla karşılaştırma
- Uyumsuzlukları olay olarak raporlama ve bu uyumsuzlukların nedenini bulacak şekilde analiz etme (örn. uyumsuzluğun kodda bir hatadan mı kaynaklandığı, test verisinde bir yanlışlık mı olduğu, test dokümanının veya testin yürütülme şeklinin yanlış mı olduğunu araştırma)
- Her bir uyumsuzluk için gerçekleştirilen düzeltmenin testini tekrarlama. Örneğin düzeltmenin doğrulanması için daha önce başarısız olmuş bir testi yeniden yürütme (onaylama testi), yazılımın değiştirilmemiş alanlarında hatalar oluşup oluşmadığından veya hata düzeltme işleminin diğer hataları ortaya çıkarmadığından emin olmak için düzeltilmiş bir testin ve/veya testlerin yürütülmesi (regresyon)

1.4.4 Çıkış Kriterlerini Değerlendirme ve Raporlama (K1)

Çıkış kriterini değerlendirme işleminde test yürütmenin belirlenen hedeflere ulaşıp ulaşamadığı değerlendirilir. Bu işlemin her bir test seviyesi için gerçekleştirilmesi gerekir (bkz. Kısım 2.2).

Çıkış kriterinin değerlendirilmesi aşamasında aşağıdaki temel adımlar yer alır:

- Test kayıtlarını, test planlamada belirtilen çıkış kriterlerine göre kontrol etme
- Daha fazla testin gerekip gerekmediğini veya belirtilen çıkış kriterinin değiştirilmesi gerekip gerekmediğini değerlendirme
- Paydaşlar için test özet raporu yazma

1.4.5 Test Kapanışı Aşaması (K1)

Test kapanışı aşamasında test projesinde elde edilen tecrübe, geliştirilen test yazılımları, metrikler ve net bilgiler bir sonraki projelerde kullanılmak için toplanır. Test kapanışı adımları, bir yazılımın piyasaya sürülmesi, test projesinin tamamlanması (veya iptal edilmesi), bir kilometre taşına ulaşılması ya da bakım sürümünün tamamlanması gibi karar verme aşamalarında hayata geçirilir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Test kapanışı işlemleri aşağıdaki temel işlemleri içerir:

- Test projesinde elde edilmesi planlanan çıktıların hangilerine ulaşıldığının kontrol edilmesi
- Açık olay raporlarını kapatma, kapatılamayanlar için değişiklik kayıtlarını oluşturma
- Yazılımın kabulünü belgeleme
- Daha sonra tekrar kullanmak amacıyla test yazılımını, test ortamını ve test altyapısını sonlandırma ve arşivleme
- Test yazılımını bakım ekiplerine devretme
- Gelecekteki sürümler ve projeler için gerekli olan değişiklikleri belirlemek amacıyla elde edilen tecrübeleri analiz etme
- Test sürecinin olgunluğunu artırmak için toplanan bilgileri kullanma

1.5 Test Etme Psikolojisi (K2)	*25 dakika*

Terimler

Hata tahminleme, bağımsızlık

Temel Bilgi

Test ve gözden geçirme sırasında sahip olunması gereken anlayış, yazılımı geliştirirken kullanılandan farklıdır. Test uzmanlarıyla aynı anlayışa sahip olan yazılımcılar, kendi kodlarını test edebilir. Ancak bu sorumluluğun bir test uzmanına verilmesinin amacı özelleşmeyi teşvik etmek ve eğitimli, profesyonel test uzmanları tarafından bağımsız bir görüş almaktır. Bağımsız testler, tüm test seviyelerinde gerçekleştirilebilir.

Belirli bir derecede bağımsızlık genellikle test uzmanının hataları ve arızaları bulma konusunda daha etkili olmasını sağlar. Ancak bir şeyi iyi bilmek veya ona aşina olmak, bağımsız olmak için bir engel teşkil etmez; bu nedenle yazılımcılar da kendi kodlarında birçok hatayı verimli bir şekilde bulabilir. Çeşitli bağımsızlık seviyeleri, aşağıda en düşükten yükseğe doğru tanımlanmıştır:

- Yazılımın kodu yazan yazılımcılar tarafından test edilmesi (en düşük bağımsızlık seviyesi)
- Yazılımın yazılım ekibindeki başka yazılımcılar tarafından test edilmesi
- Yazılımın yazılım ekibinden farklı bir ekip tarafından test edilmesi (örn. bağımsız bir test ekibi)
- Yazılımın şirket dışındaki bir ekip veya başka bir şirket (örn. test dış kaynak kullanımı hizmeti sağlayan bir şirket) tarafından test edilmesi (en yüksek bağımsızlık seviyesi)

Kişiler ve projeler, hedeflere göre hareket eder. Kişiler planlarını; yönetim ve diğer paydaşların belirlediği, hataları bulmak veya yazılımın hedefleri karşıladığını doğrulamak gibi hedeflere ulaşmaya yönelik yaparlar. Bu nedenle yapılacak bir testin hedeflerini açıkça belirtmek çok önemlidir.

Test esnasında hata bulmak, yazılıma veya yazılımı geliştiren ekibe karşı bir eleştiri olarak algılanabilir. Sonuç olarak test, yazılımın risklerinin canlıya çıkmadan önce bertaraf edilmesi açısından çok faydalı olsa da sıklıkla şirket bünyesinde yıkıcı bir işlem olarak görülür. Bir yazılımda hata aramak, merak, profesyonel kötümserlik, eleştirel bakış, detaylara dikkat etme, yazılımcılarla iyi iletişim ve hata tahminlemenin dayandırılacağı bir yaklaşım gerektirir.

Hatalar yapıcı bir yolla iletilirse, test uzmanları ile proje paydaşları arasında kötü iletişimlerin yaşanmasına engel olunur.

Test uzmanının ve test liderinin hatalar, ilerleme ve yazılımdaki riskler hakkındaki bilgileri yapıcı bir şekilde karşı tarafa iletme becerilerine sahip olması gerekir. Hatalarla ilgili bilgiler yazılımcının becerilerini geliştirmesine yardımcı olabilir. Test sırasında bulunan ve düzeltilen hatalar, zamandan ve paradan tasarruf sağlar ve riskleri azaltır.

Özellikle test uzmanları yalnızca hatalarla ilgili istenmeyen haberleri getiren bir elçi olarak görülürse iletişim problemleri meydana gelebilir. Ancak test uzmanları ve diğer paydaşlar arasındaki iletişimi ve ilişkileri geliştirmenin birçok yolu vardır:

- Karşınıza almak yerine iş birliği yapmakla başlayın, herkese daha kaliteli yazılımlar elde etme konusunda ortak bir amacınızın olduğunu hatırlatın
- Yazılımla ilgili bulgularınızı tarafsız, gerçeklere dayanır bir şekilde, yazılımı geliştiren kişiyi eleştirmeden iletin, örneğin objektif ve ekran görüntülerine dayalı olay raporları ve gözden geçirme bulguları yazın
- Karşınızdaki kişinin ne hissedeceğini ve neden tepki verdiğini anlamaya çalışın
- Karşınızdaki kişinin söylediklerinizi anladığını ve sizin de karşınızdakinin söylediklerini anladığınızı doğrulayın

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

1.6 Etik Kuralları	*10 dakika*

Yazılım testine dahil olmak, test uzmanının müşterilerin gizli ve özel bilgilerine ulşamasını sağlayabilir. Bu bilgilerin uygunsuz bir şekilde kullanılmaması için etik kurallarına uyulması gerekir. Mühendislere yönelik ACM ve IEEE etik kurallarını tanıyan ISTQB, ISTQB sertifikalı test uzmanlarının, test liderlerinin ve test yöneticilerinin aşağıdaki etik kurallara uymaları gerektiğini beyan etmektedir:

KAMU - Test uzmanları, kamu yararını gözeterek hareket etmelidir

MÜŞTERİ VE İŞVEREN - Test uzmanları, kamu yararını gözeterek, müşteri ve işverenlerinin çıkarlarına en uygun şekilde hareket etmelidir

ÜRÜN - Test uzmanları, sağladıkları çıktıların mümkün olan en yüksek profesyonel standartları karşıladığından emin olmalıdır

KARAR - Test uzmanları, profesyonel kararlarında tutarlılıklarını ve bağımsızlıklarını korumalıdır

YÖNETİM - Test yöneticileri ve liderleri, test projelerinin yönetiminde etik bir yaklaşım benimsemeli ve uygulamalıdır

MESLEK - Test uzmanları, kamu yararını gözeterek mesleğin dürüstlüğünü ve itibarını korumalıdır

MESLEKTAŞLAR - Test uzmanları, meslektaşlarına karşı adil olmalı, onları desteklemeli ve yazılımcılarla işbirliklerini geliştirmelidir

SÜREKLİ GELİŞİM - Test uzmanları, mesleklerinin icrasıyla ilgili yaşam boyu öğrenmeli ve bu konuda etik bir yaklaşım benimsemelidir

Referanslar

1.1.5 Black, 2001, Kaner, 2002
1.2 Beizer, 1990, Black, 2001, Myers, 1979
1.3 Beizer, 1990, Hetzel, 1988, Myers, 1979
1.4 Hetzel, 1988
1.4.5 Black, 2001, Craig, 2002
1.5 Black, 2001, Hetzel, 1988

2. Yazılım Yaşam Döngüsü Boyunca Test (K2)	115 dakika

Yazılım Yaşam Döngüsü Boyunca Test için Öğrenme Hedefleri

Hedefler, her bir modülü tamamladıktan sonra neler yapabileceğinizi tanımlar.

2.1 Yazılım Geliştirme Modelleri (K2)

ÖH-2.1.1 Projeyi ve yazılım çeşitlerini içeren örnekler vererek yazılım geliştirme yaşam döngüsü bünyesinde bulunan geliştirme, test ve çıktılar arasındaki ilişkiyi açıklama (K2)

ÖH-2.1.2 Yazılım geliştirme modellerinin, proje bağlamına ve yazılım karakteristiklerine göre uyarlanması gerektiğini bilme (K1)

ÖH-2.1.3 Tüm yaşam döngüsü modelleri için geçerli olan başarılı test adımlarının karakteristiklerini hatırlama (K1)

2.2 Test Seviyeleri (K2)

ÖH-2.2.1 Farklı test seviyelerini karşılaştırma: hedefler, teste tabi tutulan nesneler, test amaçları (örn. fonksiyonel veya yapısal), test çıktıları, testi gerçekleştiren kişiler, ortaya çıkarılacak hata ve arıza çeşitleri (K2)

2.3 Test Çeşitleri (K2)

ÖH-2.3.1 Dört yazılım testi çeşidini (fonksiyonel, fonksiyonel olmayan, yapısal ve değişiklikle ilgili) örneklerle karşılaştırma (K2)

ÖH-2.3.2 Fonksiyonel ve yapısal testlerin tüm test seviyelerinde yapılabildiğini bilme (K1)

ÖH-2.3.3 Fonksiyonel olmayan gereksinimleri baz alarak fonksiyonel olmayan test çeşitlerini tanımlama ve açıklama (K2)

ÖH-2.3.4 Bir yazılımın iç yapısının veya mimarisinin analizine göre test çeşitlerini tanımlama ve açıklama (K2)

ÖH-2.3.5 Onaylama testi ve regresyon testinin amacını açıklama (K2)

2.4 Bakım Testi (K2)

ÖH-2.4.1 Bakım testini (var olan bir sistemi test etme) yeni bir sistemi test etme ile karşılaştırma. Karşılaştırmayı yaparken test çeşitlerini, test tetikleyicileri ve test eforunu dikkate alma (K2)

ÖH-2.4.2 Bakım testine yönelik göstergeleri bilme (modifikasyon, taşıma ve kullanımdan çıkarma) (K1)

ÖH-2.4.3. Bakımda regresyon testinin ve etki analizinin rolünü açıklama (K2)

2.1 Yazılım Geliştirme Modelleri (K2)	*20 dakika*

Terimler

Ticari Paket Yazılım (COTS), döngüsel-artımlı geliştirme modeli, sağlama, doğrulama, V modeli

Temel Bilgi

Test tek başına var olamaz, test adımları yazılım geliştirme adımları ile iç içe geçmiş durumdadır. Farklı yazılım geliştirme yaşam döngüsü modelleri, farklı test yaklaşımları gerektirir.

2.1.1 V modeli (Sıralı Geliştirme Modeli) (K2)

V modelinin farklı çeşitleri olsa da en yaygın kullanılan V modeli çeşidi, dört geliştirme seviyesine karşılık gelen dört test seviyesinin kullanımıdır.

Bu ders programında ele alınan dört test seviyesi şöyledir:

- Bileşen (birim) testi
- Entegrasyon testi
- Sistem testi
- Kabul testi

Uygulamada, projeye ve yazılıma bağlı olarak V modelinin daha fazla, daha az veya farklı geliştirme ve test seviyeleri olabilir. Örneğin, bileşen testinden sonra bileşen entegrasyon testi, sistem testinden sonra sistem entegrasyon testi olabilir.

Yazılım geliştirme sırasında üretilen ürünler (iş senaryoları veya kullanım senaryoları, gereksinimler, tasarım belgeleri ve kod vb) genellikle bir veya daha fazla test seviyesinde test esasını oluşturur. Ürünlerin tanımlanmasına yönelik verilebilecek referanslar arasında Entegre Yetenek Olgunluk Modelini (CMMI) veya "Yazılım yaşam döngüsü süreçlerini" (IEEE/IEC 12207) sayabiliriz. Doğrulama ve sağlama (ve erken test tasarımı), yazılım ürünlerinin geliştirilmesi sırasında da gerçekleştirilebilir.

2.1.2 Döngüsel-Artımlı Geliştirme Modelleri (K2)

Döngüsel-artımlı geliştirme, kısa geliştirme döngüleri boyunca gereksinimleri çıkarma, yazılımı tasarlama, geliştirme ve test etme sürecidir. Örnek olarak Hızlı Uygulama Geliştirme (RAD), Rational Unified Process (RUP) ve çevik geliştirme modelleri verilebilir. Bu modeller kullanılarak üretilen bir yazılım, her bir döngü sırasında çeşitli test seviyelerinde test edilebilir. Daha önce geliştirilen kısma eklenen bir aşama, büyüyen kısmi bir sistem oluşturur ve bu sistemin de test edilmesi gerekir. Regresyon testi, ilk döngüden başlayarak sonraki tüm döngülerde artan bir önem taşır. Doğrulama ve sağlama her bir aşamada gerçekleştirilebilir.

2.1.3 Yaşam Döngüsü Modeli İçinde Test (K2)

Tüm yaşam döngüsü modellerinde iyi test etme pratiğinin birtakım özellikleri vardır:

- Her geliştirme aktivitesi için buna karşılık gelen bir test etme aktivitesi vardır
- Her bir test seviyesinde bu seviyeye özgü test hedefleri bulunur
- Belirli bir test seviyesi için testlerin analizi ve tasarımı karşılık gelen geliştirme akivitesi sırasında başlamalıdır
- Dokümanlar belli bir olgunluğa geldiği anda test uzmanları tarafından gözden geçirilmelidir

Projenin doğasına veya yazılım mimarisine bağlı olarak test seviyeleri birleştirilebilir veya yeniden düzenlenebilir. Örneğin, ticari bir paket yazılımın (COTS) bir sisteme entegre edilmesi için, sistem seviyesinde entegrasyon testi (örn. altyapıya veya diğer sistemlere entegrasyon ya da sistem dağıtımı) ve kabul testi (fonksiyonel ve/veya fonksiyonel olmayan ve kullanıcı ve/veya operasyonel test) gerçekleştirilebilir.

2.2 Test Seviyeleri (K2)	*40 dakika*

Terimler

Alfa testi, beta testi, bileşen testi, sürücü, saha testi, fonksiyonel gereksinim, entegrasyon, entegrasyon testi, fonksiyonel olmayan gereksinim, sağlamlık testi, taklit uygulama, sistem testi, test ortamı, test seviyesi, test güdümlü geliştirme, kullanıcı kabul testi

Temel Bilgi

Test seviyelerinin her biri için şunlar belirlenebilir: genel hedefler, test senaryoları türetmek için referans alınan çalışma ürünleri (örn. test esası), test nesnesi (örn. test edilen şey), bulunacak genel hatalar ve arızalar, test kuluçkası gereksinimleri, araç desteği, özel yaklaşımlar ve sorumluluklar.

Bir sistemin yapılandırma verilerinin test edilmesi, test planlama sırasında ele alınmalıdır.

2.2.1 Bileşen (Birim) Testi (K2)

Test esası:

- Bileşen gereksinimleri
- Ayrıntılı tasarım
- Kod

Kullanılan genel test nesneleri:

- Bileşenler
- Programlar
- Veri dönüştürme / taşıma programları
- Veritabanı modülleri

Bileşen testi (birim, modül veya program testi olarak da bilinir), ayrı ayrı test edilebilen yazılım modüllerindeki, programlardaki, nesnelerdeki, sınıflardaki vb. hataları arar ve bunların işleyişini doğrular. Yazılım geliştirme yaşam döngüsü ve sistemin bağlamına göre sistemin geri kalanından bağımsız şekilde gerçekleştirilebilir. Taklit uygulamalar, sürücüler ve simülatörler kullanılabilir.

Bileşen testi; fonksiyonalite testini, fonksiyonel olmayan testleri, yapısal testleri veya sağlamlık testlerinden oluşabilir. Test senaryoları, bileşenin gereksinimi, yazılım tasarımı veya veri modeli gibi test esaslarından türetilebilir.

Bileşen testinde genellikle test edilen koda erişim sözkonusudur. Aynı zamanda, birim testi çerçevesi ya da hata ayıklama aracı gibi bir geliştirme ortamının desteğiyle gerçekleştirilir. Uygulamada, bileşen testi genellikle kodu yazan programcı tarafından yapılır. Bileşen testlerinde çoğunlukla hatalar bulundukları anda kayıt altına alınmadan düzeltilir.

Bileşen testine yönelik yaklaşımlardan biri, kodlamadan önce test senaryolarını hazırlamak ve otomasyona geçirmektir. Bu yaklaşıma test öncelikli yaklaşım veya test güdümlü geliştirme adı verilir ve oldukça döngüseldir. Test senaryolarını geliştirme, ardından küçük kod parçalarını oluşturma ve entegre etme ardından da tüm sorunları düzelterek ve başarılı olana kadar yineleyerek bileşen testlerini yürütme döngüsüne dayanır.

2.2.2 Entegrasyon Testi (K2)

Test esası:

- Yazılım ve sistem tasarımı
- Mimari
- İş akışları
- Kullanım senaryoları

Kullanılan genel test nesneleri:

- Alt sistemler
- Veritabanı uyarlama
- Altyapı
- Arayüzler
- Sistem yapılandırması ve yapılandırma verisi

Entegrasyon testi, bileşenler arasındaki arayüzleri, yazılımın işletim sistemi, dosya sistemi ve donanım gibi sistemin farklı bölümleriyle etkileşimlerini ve sistemler arasındaki arayüzleri test eder.

Birden fazla entegrasyon testi seviyesi olabilir ve aşağıdaki gibi çeşitli boyutlardaki test nesneleri üzerinde gerçekleştirilebilir:

1. Bileşen entegrasyon testi, yazılımın bileşenleri arasındaki etkileşimleri test eder ve bileşen testinden sonra yapılır.
2. Sistem entegrasyon testi, farklı sistemler veya donanım ve yazılım arasındaki etkileşimleri test eder ve sistem testinden sonra yapılabilir.

Entegrasyonun kapsamı ne kadar geniş olursa, hataların belirli bir bileşene veya sisteme indirgenmesi de o kadar zor olur, bu durumda daha fazla risk ortaya çıkabilir ve sorun gidermeye harcanan süre artabilir.

Sistematik entegrasyon stratejileri, sistem mimarisine (yukarıdan aşağıya veya aşağıdan yukarıya gibi), fonksiyonel görevlere, işlemleri ele alma sıralarına veya sistem ya da bileşenlerle ilgili diğer bazı kapsamlara dayanabilir. Kusur izolasyonunu kolaylaştırmak ve hataları erken saptamak için entegrasyonun "büyük patlama" yerine aşamalı olması gerekir.

Fonksiyonel olmayan karakteristiklerin (örn. performans) test edilmesi işlemi, fonksiyonel teste dahil edilebileceği gibi entegrasyon testine de dahil edilebilir.

Her bir entegrasyon aşamasında test uzmanları yalnızca entegrasyona odaklanır. Örneğin, modül A'yı modül B'ye entegre ediyorlarsa, her bir modülün fonksiyonalitesini değil sadece bu modüller arasındaki iletişimi test etmelidirler.

İdeal yaklaşım, test uzmanlarının mimariyi anlayıp entegrasyon testi kurgusunu planlamalarıdır. Örneğin, entegrasyon testleri bileşenlerin veya sistemlerin oluşturulmasından önce planlandıysa, bu bileşenlerin entegrasyon testlerini en etkin şekilde yapabilmek için gereken sırada oluşturulmasını sağlayabilirler.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

2.2.3 Sistem Testi (K2)

Test esası:
- Sistem ve yazılım gereksinimleri
- Kullanım senaryoları
- Fonksiyonel gereksinimler
- Risk analizi raporları

Kullanılan genel test nesneleri:
- Sistem, kullanıcı ve operasyon kılavuzları
- Sistem yapılandırması

Sistem testi, tüm sistemin/yazılımın davranışı ile ilgilidir. Master test planında veya ilgili test seviyesinin planında sistem testinin kapsamı açıkça belirtilir.

Sistemin çalışacağı ortamla ilgili hata riskini en aza indirmek için sistem testinde test ortamı mümkün olduğunca canlı ortama yakın olmalıdır.

Sistem testi, risklere ve/veya gereksinimlere, iş süreçlerine, kullanım senaryolarına veya sistem davranışını tanımlayan metinlere ya da modellere, işletim sistemiyle etkileşimlere ve sistem kaynaklarına dayanabilir.

Sistem testi, sistemin fonksiyonel ve fonksiyonel olmayan gereksinimlerini ve veri kalitesini sorgulamalıdır. Bu seviyede test uzmanlarının tamamlanmamış gereksinimlerle de ilgilenmesi gerekir. Fonksiyonel gereksinimlerle ilgili sistem testi, test edilecek sistem için en uygun gereksinim bazlı (kara kutu) teknikler kullanılarak başlar. Örneğin, iş kurallarında tanımlanan etkilerin kombinasyonları için karar tablosu test tekniği kullanılabilir. Ardından, yapısal testlere (beyaz kutu) geçilerek gereksinim bazlı testlerin yakalayamadığı hatalar yakalanabilir. Örneğin menü yapısı ve web sayfası navigasyonunu test nesnesi olarak ele alan yapısal testler. (bkz. Bölüm 4).

Sistem testi genellikle bağımsız bir test ekibi tarafından gerçekleştirir.

2.2.4 Kabul testi (K2)

Test esası:
- Kullanıcı gereksinimleri
- Sistem gereksinimleri
- Kullanım senaryoları
- İş süreçleri
- Risk analizi raporları

Kullanılan genel test nesneleri:
- İş süreçleri
- Operasyonel süreçler ve bakım süreçleri
- Kullanıcı prosedürleri
- Formlar
- Raporlar
- Yapılandırma verisi

Kabul testi genellikle bir yazılımın müşterilerinin veya kullanıcılarının sorumluluğundadır; bu seviyedeki testlere diğer paydaşlar da dahil olabilir.

Kabul testinin amacı, sisteme, sistemin parçalarına veya sistemin fonksiyonel olmayan gereksinimlerine karşı güven oluşturmaktır. Kabul testinde ana odak hataları bulmak değildir, sistemin canlıya hazır olduğunu göstermektir. Kabul testinin son test seviyesi olmadığı durumlar olabilir buna rağmen kabul testinde sistemin canlıya alınmaya ve kullanıma hazır olup olmadığı denetlenebilir. Örneğin, geniş ölçekli sistem entegrasyon testi, kabul testinin ardından yapılabilir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Kabul testi yazılım geliştirme yaşam döngüsünde birçok kez yapılabilir, örneğin:
- o Bir paket yazılım kurulduğunda veya entegre edildiğinde kabul testinden geçebilir
- o Bir bileşenin kullanılabilirliği ile ilgili kabul testi, bileşen testi sırasında yapılabilir
- o Yeni bir fonksiyonel geliştirme ile ilgili kabul testi, sistem testinden önce yapılabilir

Genel kabul testi biçimleri aşağıdaki gibidir:

Kullanıcı kabul testi
Sistemin son kullanıcılar tarafından kullanımının uygun olduğunu doğrular.

Operasyonel (kabul) test
Sistemin, sistem yöneticileri tarafından kabulüdür; şunları içerir:
- Yedekleme/geri yükleme testi
- Felaket kurtarma
- Kullanıcı yönetimi
- Bakım görevleri
- Veri yükleme ve veri göçü görevleri
- Güvenlik açıklarının periyodik kontrolleri

Sözleşme ve yasal mevzuata göre yapılan kabul testleri
Sözleşmeye göre yapılan kabul testi, müşteriye özel geliştirilen yazılımın sözleşmenin şartlarına göre gerçekleştirilir. Kabul kriteri, taraflar sözleşmeyi kabul ettiğinde belirlenmelidir. Mevzuata göre yapılan kabul testi, devletin belirlediği, yasal veya emniyetle ilgili düzenlemeler gibi uyulması gereken mevzuatlara göre gerçekleştirilir.

Alfa ve beta (veya saha) testi
Paket yazılım geliştiren şirketler geliştirdikleri yazılımı satış için pazara sunmadan önce potansiyel veya var olan müşterilerinden geri bildirim almak isterler. Alfa testi, yazılımı geliştiren şirketin kendi bünyesinde kontröllü bir şekilde yapılırken beta testi veya saha testi, müşterilerin veya potansiyel müşterilerin kendi ortamlarında kontrolsüz bir şekilde gerçekleştirilir.

Terminolojide beta testleri için fabrika kabul testi veya saha kabul testi gibi terimler de kullanılmaktadır.

2.3 Test Çeşitleri (K2)	*40 dakika*

Terimler

Kara kutu testi, kod kapsamı, fonksiyonel test, birlikte çalışabilirlik testi, yükleme testi, sürdürülebilirlik testi, performans testi, taşınabilirlik testi, güvenilirlik testi, güvenlik testi, stres testi, yapısal test, kullanılabilirlik testi, beyaz kutu testi

Genel Bilgi

Özel bir nedene veya test hedefine dayanarak yazılımın (veya yazılımın bir bölümünün) testi yapılabilir.

Test çeşidi belirli bir test hedefine odaklanır, bu hedef aşağıdakilerden herhangi biri olabilir:

- Yazılımın gerçekleştireceği bir fonksiyon
- Güvenilirlik veya kullanılabilirlik gibi fonksiyonel olmayan gereksinimler
- Yazılımın veya sistemin yapısı ya da mimarisi
- Değişiklik ile ilgili. Örn. hataların düzeltildiğini onaylama (onaylama testi) ve istenmeyen değişiklikleri arama (regresyon)

Yapısal test (örn. kontrol akışı modeli veya menü yapısı modeli), fonksiyonel olmayan test (örn. performans modeli, kullanılabilirlik modeli, güvenlik tehdidi modellemesi) ve fonksiyonel test için (örn. süreç akış modeli, durum geçişi modeli) yazılıma benzeyen bir model geliştirilebilir ve/veya kullanılabilir.

2.3.1 Fonksiyonu Test Etme (Fonksiyonel Test) (K2)

Bir sistemin, alt sistemin veya bileşenin gerçekleştirmesi gereken fonksiyonlar, gereksinimler, kullanım senaryoları veya bir fonksiyonel spesifikasyon gibi kriterler için tanımlanabilir. Fonksiyonlar, sistemin "yaptıklarıdır".

Fonksiyonel testler, fonksiyonlara ve özelliklere (dokümanlarda tanımlanan veya test uzmanları tarafından görüşmeler sonucunda elde edilmiş) ve bunların belirli sistemlerle birlikte çalışabilirliğine dayanır. Tüm test seviyelerinde gerçekleştirilebilir.

Yazılımın fonksiyonalitesinden gereksinim bazlı test teknikleri kullanılarak test koşulları ve test senaryoları türetilebilir (bkz. Bölüm 4). Fonksiyonel testlerde genellikle yazılımın harici davranışları, girdi ve çıktılar dikkate alınır. (kara kutu testi).

Bir çeşit fonksiyonel test olan güvenlik testi, kötü amaçlı dış kaynaklardan gelen tehditlerin algılanması ile ilgili fonksiyonları ele alır (örn. güvenlik duvarı). Fonksiyonel testin diğer bir çeşidi olan birlikte çalışabilirlik testi, yazılımın belirtilen bir veya daha fazla bileşenle veya sistemle etkileşim kurma yeteneğini değerlendirir.

2.3.2 Fonksiyonel Olmayan Gereksinimleri Test Etme (Fonksiyonel Olmayan Test) (K2)

Fonksiyonel olmayan test, performans, yükleme, stres, kullanılabilirlik, sürdürülebilirlik, güvenilirlik ve taşınabilirlik gibi testleri içerir fakat bunlarla sınırlı değildir. Bu, yazılımın "nasıl" çalıştığını gösteren bir testtir.

Fonksiyonel olmayan test, tüm test seviyelerinde gerçekleştirilebilir. Fonksiyonel olmayan test terimi, yazılımın performans testindeki yanıt süreleri gibi değişen bir skalada ölçülebilen karakteristiklerini ölçmek için gereken testleri tanımlar. Bu testler, "Yazılım Mühendisliği – Yazılım Kalitesi" (ISO 9126) kapsamında tanımlanan model gibi bir kalite modelini referans alabilir. Fonksiyonel olmayan test, genellikle yazılımın harici davranışını göz önünde bulundurur ve bunu gerçekleştirmek için çoğu test senaryosunda kara kutu test tasarım tekniklerini kullanır.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

2.3.3 Yazılım Yapısını/Mimarisini Test Etme (Yapısal Testler) (K2)

Yapısal (beyaz kutu) testler, tüm test seviyelerinde gerçekleştirilebilir. Yapısal tekniklerin test kapsamını tamamlamaya yardımcı olması amacıyla gereksinim bazlı tekniklerden sonra kullanılması önerilir.

Test kapsamı, yapının bir test grubu tarafından çalıştırılma derecesidir ve kapsanan öğelerin yüzdesi olarak ifade edilir. Kapsam %100 değilse, kapsamı artırmak amacıyla test edilmeyen öğeleri test etmek için daha fazla test tasarlanabilir. Kapsam teknikleri Bölüm 4'te yer almaktadır.

Bileşen testi ve bileşen entegrasyon testi başta olmak üzere tüm test seviyelerinde, komut ve karar öğelerinin kod kapsamını ölçmek için araçlar kullanılabilir. Yapısal testler, bir çağırma hiyerarşisi gibi sistem mimarisine dayanabilir.

Yapısal test yaklaşımları ayrıca sistem, sistem entegrasyonu veya kabul testi seviyelerinde uygulanabilir (örn. menü yapıları).

2.3.4 Değişiklikleri Test Etme: Tekrar Testi ve Regresyon (K2)

Bir hata tespit edildikten ve düzeltildikten sonra bulunan hatanın başarılı şekilde ortadan kaldırıldığını onaylamak için yazılım yeniden test edilmelidir. Bu işleme onay adı verilir. Hata ayıklama (bir hatayı bulma ve düzeltme) bir geliştirme işlemidir, test işlemi değildir.

Regresyon, yapılan değişiklikler sonucunda oluşan yeni hataları keşfetmek amacıyla zaten test edilmiş olan bir programı yeniden test etme işlemidir. Yeni oluşan hatalar, test edilmekte olan yazılımda veya doğrudan veya dolaylı bir başka yazılım bileşenin de olabilir. Yazılım veya yazılımın ortamı değiştirildiğinde gerçekleştirilir. Regresyonun kapsamı, daha önceden çalışan yazılımdaki hataları bulamama riskine dayanır.

Yazılan test senaryoları, onaylama testinde kullanılması ve regresyona yardımcı olması amacıyla tekrar edilebilir olmalıdır.

Regresyon, tüm test seviyelerinde gerçekleştirilebilir ve fonksiyonel, fonksiyonel olmayan ve yapısal testleri içerir. Regresyon test grupları birçok kez çalıştırılır ve genellikle yavaş bir şekilde ilerler; bu nedenle regresyon, otomasyon için güçlü bir adaydır.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

2.4 Bakım Testi (K2)	*15 dakika*

Terimler

Etki analizi, bakım testi

Genel Bilgi

Bir yazılım piyasaya sürüldükten veya canlıya alındıktan sonra genellikle yıllarca veya on yıllarca kullanılır. Bu süre zarfında yazılım üzerinde iyileştirmeler ve yeni değşiklikler yapılarak düzeltilir, değiştirilir veya geliştirilir. Başarılı bir bakım testi için sürümlerin önceden planlanması çok önemlidir. Planlanan sürümler ve düzeltmeler arasındaki farkın belirgin olması gerekmektedir. Bakım testi, varolan bir operasyonel sistem üzerinde gerçekleştirilir ve modifikasyon, taşıma veya yazılımın kullanımdan kalkması ile tetiklenir.

Modifikasyonlar; işletim sisteminin yeni ortaya çıkan veya yeni keşfedilen güvenlik açıklarını düzeltmek amacıyla planlı işletim sistemi veya veritabanı yükseltmeleri, planlı ticari paket yazılım yükseltmesi veya sunulan yamalar gibi planlı geliştirme değişikliklerini (örn. sürüm tabanlı), düzeltici değişiklikleri, acil durum değişikliklerini ve ortam değişikliklerini içerir.

Taşıma (bir platformdan diğerine) için bakım testi, değiştirilen yazılımda olduğu kadar yeni ortamda da gerçekleştirilmesi gereken operasyonel testleri içermelidir. Taşıma testi (dönüşüm testi), başka bir uygulamadaki veriler bakımı yapılan sisteme taşınacağı zaman da gereklidir.

Bir sistemin kullanımdan kaldırılmasına yönelik bakım testi, veri taşıma testini ve uzun süreli veri saklama dönemleri gerekmesi durumunda arşivlemeyi içerebilir.

Değişikliklerle ilgili testlere ek olarak bakım testi, değiştirilmeyen sistem bölümlerinde gerçekleştirilen regresyonu da içerir. Bakım testinin kapsamı, değişiklik riskine, var olan sistemin boyutuna ve değişikliğin boyutuna göre değişir. Değişikliklere bağlı olarak bakım testi, tüm test seviyelerinde ve tüm test çeşitleri için gerçekleştirilebilir. Var olan sistemin değişikliklerden ne şekilde etkileneceğini belirlemeye etki analizi adı verilir ve bu analiz ne kadar regresyonun yapılacağına karar vermede kullanılır. Etki analizi, regresyon test grubunu belirlemek için kullanılabilir.

Gereksinimler güncel değilse veya eksikse ya da alan bilgisine sahip test uzmanları yoksa bakım testi zor olabilir.

Referanslar

2.1.3 CMMI, Craig, 2002, Hetzel, 1988, IEEE 12207 2.2 Hetzel, 1988
2.2.3 Copeland, 2004, Myers, 1979

2.3.1 Beizer, 1990, Black, 2001, Copeland, 2004
2.3.2 Black, 2001, ISO 9126
2.3.3 Beizer, 1990, Copeland, 2004, Hetzel, 1988
2.3.4 Hetzel, 1988, IEEE STD 829-1998
2.3.5 2.4 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE STD 829-1998

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

3. Statik Teknikler (K2)	60 dakika

Statik Teknikler için Öğrenme Hedefleri

Hedefler, her bir modülü tamamladıktan sonra neler yapabileceğinizi tanımlar.

3.1 Statik Teknikler ve Test Süreci (K2)

ÖH-3.1.1 Farklı statik tekniklerin uygulanabileceği yazılımları tanıma (K1)

ÖH-3.1.2 Yazılımın değerlendirilmesi için statik teknikleri kullanmanın önemini ve değerini açıklama (K2)

ÖH-3.1.3 Hedefleri, hata çeşitlerini ve tekniklerin yazılım yaşam döngüsündeki rolünü göz önünde bulundurarak statik ve dinamik teknikler arasındaki farkı açıklama (K2)

3.2 Gözden Geçirme Süreci (K2)

ÖH-3.2.1 Genel bir resmi gözden geçirmenin işlemlerini, rollerini ve sorumluluklarını hatırlama (K1)

ÖH-3.2.2 Farklı gözden geçirme çeşitleri arasındaki farkları açıklama: gayri resmi gözden geçirme, teknik gözden geçirme, üzerinden geçme ve teftiş (K2)

ÖH-3.2.3 Başarılı gözden geçirme sürecinin faktörlerini açıklama (K2)

3.3 Araçlarla Gerçekleştirilen Statik Analiz (K2)

ÖH-3.3.1 Statik analiz ile bulunan hataları hatırlama ve bunları gözden geçirmeler ve dinamik test ile karşılaştırma (K1)

ÖH-3.3.2 Örnekler kullanarak statik analizin genel avantajlarını açıklama (K2)

ÖH-3.3.3 Statik analiz araçları ile tanımlanabilen genel kod ve tasarım hatalarını listeleme (K1)

3.1 Statik Teknikler ve Test Süreci (K2)	*15 dakika*

Terimler

Dinamik test, statik test

Genel Bilgi

Yazılımın yürütülmesini gerektiren dinamik testin aksine statik test teknikleri, kod yürütülmeden kodun veya diğer proje dokümanlarının manuel olarak incelenmesine (gözden geçirme) ve otomatik şekilde analiz edilmesine (statik analiz) dayanır.

Gözden geçirme, yazılımı (kod dahil) test etmenin bir yoludur ve dinamik testler yapılmadan önce gerçekleştirilebilir. Yazılım geliştirme yaşam döngüsünün başlarında gözden geçirmeler sırasında tespit edilen hataları ortadan kaldırmak (örn. gereksinimlerde bulunan hatalar) genellikle yürütülen kod üzerinde çalıştırılan testlerle tespit edilen hataları ortadan kaldırmaktan daha ucuzdur.

Gözden geçirme, tamamen manuel bir işlem olarak gerçekleştirilebilir, ancak araç desteği de bulunur. Temel işlem, bir dokümanı incelemek ve bu doküman hakkında yorumlar yapmaktır. Gereksinimler, tasarımlar, kod, test planları, test gereksinimleri, test senaryoları, test komut dosyaları, kullanım kılavuzları veya web sayfaları gibi tüm yazılım ürünleri gözden geçirilebilir.

Gözden geçirmenin avantajları arasında, erken hata tespiti ve düzeltmesi, geliştirme sürecinde üretkenlik iyileştirmeleri, daha hızlı yazılım geliştirme, azalan test maliyeti ve süresi, yaşam boyu maliyet azalmaları, daha az hata ve gelişmiş iletişim sayılabilir. Gözden geçirmeler, dinamik testte bulunamayan gereksinim eksikliklerini bulabilir.

Gözden geçirme, statik analiz ve dinamik test aynı hedefe sahiptir: hataları tespit etme. Birbirlerini tamamlarlar, farklı teknikler farklı hata çeşitlerini etkin ve verimli bir şekilde bulabilir. Dinamik test ile karşılaştırıldığında statik teknikler, hataları değil hataların nedenlerini bulur.

Gözden geçirmelerde dinamik testlerde olduğundan daha kolay bulunan genel hatalar şunlardır: standartlardan sapma, gereksinim hataları, tasarım hataları, yetersiz sürdürülebilirlik ve yanlış arayüz gereksinimleri.

3.2 Gözden Geçirme Süreci (K2)	*25 dakika*

Terimler

Giriş kriteri, resmi gözden geçirme, gayri resmi gözden geçirme, teftiş, metrik, moderatör, eş-gözden geçirme, gözden geçirici, katip, teknik gözden geçirme, üzerinden geçme

Genel Bilgi

Gözden geçiriciler için yazılı prosedürlerin olmadığı gayri resmi gözden geçirmelerden, ekip katılımıyla gerçekleşen, gözden geçirme sonuçlarının belgelendiği ve gözden geçirmeyi gerçekleştirmek için belgelenmiş prosedürlerin olduğu sistematik gözden geçirmelere kadar farklı gözden geçirme çeşitleri mevcuttur. Gözden geçirme sürecinin resmiliği, geliştirme sürecinin olgunluğu, yasal veya düzenlemeler ile ilgili gereksinimler ya da denetim izlemesi gerekliliği gibi faktörlerle ilgilidir.

Gözden geçirmenin gerçekleştirilme şekli, gözden geçirme ile ilgili kabul edilen hedeflere göre değişir (örn. hataları bulma, konuyu kavrama, test uzmanlarını ve yeni ekip üyelerini eğitme veya ortak fikir oluşturmak için tartışma ve karar verme).

3.2.1 Resmi Gözden Geçirme İşlemleri (K1)

Genel bir resmi gözden geçirmenin temel işlemleri aşağıdaki gibidir:

1. Planlama
   - Gözden geçirme kriterini belirleme
   - Personeli seçme
   - Rolleri dağıtma
   - Daha resmi gözden geçirme çeşidi için giriş ve çıkış kriterini belirleme (örn. teftişler)
   - Belgenin hangi bölümlerinin gözden geçirileceğini seçme
   - Giriş kriterini kontrol etme
2. Başlama
   - Belgeleri dağıtma
   - Katılımcılara hedefleri, süreci ve belgeleri açıklama
3. Kişisel hazırlık
   - Belgeleri gözden geçirerek gözden geçirme toplantısına hazırlanma
   - Potansiyel hataları, soruları ve yorumları not etme
4. Sonuçları inceleme/değerlendirme/kaydetme (gözden geçirme toplantısı)
   - Belgelenen sonuçlar veya tutanaklar üzerinden tartışma veya kayıt tutma
   - Hataları not etme, hataların giderilmesiyle ilgili öneriler sunma, hatalarla ilgili kararlar verme
   - Toplantılar sırasında veya grubun elektronik iletişimlerini izleyerek sorunları inceleme/değerlendirme ve kaydetme
5. Yeniden çalışma
   - Bulunan hataları düzeltme (genellikle gözden geçirilen ürünün sahibi/yazar tarafından yapılır)
   - Hataların güncel durumunu kaydetme (resmi gözden geçirmelerde)
6. Takip
   - Hataların ele alınıp alınmadığını kontrol etme
   - Metrikleri toplama
   - Çıkış kriterini kontrol etme

3.2.2 Roller ve Sorumluluklar (K1)

Olağan bir resmi gözden geçirme toplantısına aşağıdaki kişiler katılır:

- Yönetici: Önerilen değerlendirmelerin uygulanıp uygulanmayacağına karar verir, uygulanacak önerileri proje takvimine ekler ve gözden geçirme toplantısının hedeflerine ulaşılıp ulaşılmadığına karar verir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

- Moderatör: Doküman veya doküman grubunun gözden geçirilme işlemini, gözden geçirmenin planlanmasını, toplantının gerçekleştirilmesini ve toplantı sonrasında takiplerin yapılmasını yöneten kişidir. Gerekirse moderatör çeşitli bakış açıları arasında aracılık yapabilir ve genellikle gözden geçirme başarısının bağlı olduğu kişidir.
- Yazar: Gözden geçirilecek dokümanların sahibi veya bunlardan sorumlu olan kişidir.
- Gözden geçiriciler: Gerekli hazırlıklardan sonra gözden geçirilen dokümandaki bulguları (örn. hatalar) tanımlayan ve açıklayan, teknik veya alan bilgisine sahip kişilerdir (kontrol edici veya müfettiş adı da verilir). Gözden geçiricilerin gözden geçirme sürecinde farklı bakış açılarını temsil etmesi ve tüm gözden geçirme toplantılarında yer alması gerekir.
- Katip (veya kaydedici): Toplantı sırasında tanımlanan tüm sorunları, problemleri ve açık noktaları kayıt altına alır.

Yazılıma farklı bakış açılarından bakmak ve kontrol listelerini kullanmak, gözden geçirme işlemlerinin daha verimli ve etkili olmasını sağlar. Örneğin; kullanıcı, bakım görevlisi, test uzmanı veya operasyonlar gibi çeşitli bakış açılarına dayanan bir kontrol listesi veya genel gereksinim problemlerinden oluşan bir kontrol listesi, daha önceden tespit edilmemiş sorunların gün yüzüne çıkmasını sağlayabilir.

3.2.3 Gözden Geçirme Çeşitleri (K2)

Tek bir yazılım birden fazla gözden geçirmenin konusu olabilir. Birden fazla gözden geçirme çeşidi kullanılırsa, hangisinin önce yapılacağı değişkenlik gösterebilir. Örneğin; gayri resmi gözden geçirme, teknik gözden geçirmeden önce gerçekleştirilmelidir veya bir teftiş, müşterilerle üzerinden geçmeden önce gereksinimler üzerinde yapılabilir. Yaygın gözden geçirme çeşitlerinin temel özellikleri, seçenekleri ve amaçları şöyledir:

Gayri Resmi Gözden Geçirme

- Resmi süreç yoktur
- Eşli programlama şeklinde olabilir veya tasarımları ve kodu gözden geçiren tecrübeli teknik biri tarafından gerçekleştirilebilir
- Sonuçlar kayıt altına alınması isteğe bağlıdır
- Gözden geçiricilerin yetkinliğine göre faydası değişebilir
- Temel amaç: hızlı ve kolay bir şekilde hataların bulunması

Üzerinden Geçme

- Üzerinden geçme toplantıları yazar tarafından yönetilir
- Senaryoların üzerinden geçme, prova yapma, eş grup katılımı şeklinde yapılabilir
- Açık uçlu oturumlar
    - Gözden geçiricilerin isteğine bağlı olarak toplantı öncesi hazırlığı
    - Bulgular listesini içeren gözden geçirme raporunun isteğe bağlı hazırlanması
- Katip kullanımı isteğe bağlıdır. Katip, gözden geçirilecek iş ürünün sahibi olmamalıdır.
- Üzerinden geçme süreci gayri resmiden çok resmiye kadar değişebilir
- Temel amaçlar: öğrenme, anlama, hataları bulma

Teknik Gözden Geçirme

- Dokümante edilmiş, tanımlanmış hata bulma süreci olan teknik kişileri ve çalışma arkadaşlarını sürece dahil eden, isteğe bağlı olarak yönetimin de katıldığı
- Yönetim katılımı olmadan eş-gözden geçirme olarak gerçekleştirilebilir
- İdeal olarak eğitimli moderatör tarafından yönetilir
- Gözden geçiriciler tarafından toplantı öncesi hazırlığı yapılır
- İsteğe bağlı kontrol listesi kullanılır
- Bulgular listesini, yazılımın gereksinimleri karşılayıp karşılamadığı ile ilgili kararı ve uygun olan durumlarda bulgularla ilgili önerileri içeren gözden geçirme raporunun hazırlanması
- Uygulamada, oldukça gayri resmi olabileceği gibi çok resmi de olabilir
- Temel amaçlar: tartışma, karar verme, alternatifleri değerlendirme, hataları bulma, teknik problemleri çözme ve gereksinimlere, planlara, düzenlemelere ve standartlara uyumu kontrol etme

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Teftiş

- Eğitimli moderatör tarafından yönetilir
- Genellikle bir eş inceleme olarak yürütülür
- Tanımlı roller
- Metrik toplamayı içerir
- Kurallara ve kontrol listelerine dayanan resmi süreç
- Yazılım ürününün kabulü için belirli giriş ve çıkış kriteri
- Toplantı öncesi hazırlık
- Bulgular listesini içeren teftiş raporu
- Resmi takip süreci (isteğe bağlı süreç iyileştirmesi bileşenleri ile)
- İsteğe bağlı okuyucu
- Temel amaç: hataları bulmak

Üzerinden geçme, teknik gözden geçirme ve teftiş, bir eş grubu içinde gerçekleştirilebilir; örn. aynı ünvana sahip meslektaşlar. Bu gözden geçirme çeşidine "eş-gözden geçirme" adı verilir.

3.2.4 Gözden Geçirme için Başarı Faktörleri (K2)

Gözden geçirmeler için başarı faktörleri şunları içerir:

- Her bir gözden geçirmenin önceden belirlenmiş net hedefleri vardır
- Gözden geçirme hedeflerine uygun kişiler dahil edilir
- Test uzmanları, gözden geçirmeye katkı sağlayan aynı zamanda testleri daha erken hazırlamalarını sağlayacak şekilde ürün hakkında bilgi edinen gözden geçiricilerdir
- Bulunan hatalar normal karşılanır ve objektif şekilde ifade edilir
- İnsan psikolojisine dikkat edilir (örn. süreç yazar için olumlu bir deneyim haline getirilir)
- Gözden geçirme güven ortamı içinde gerçekleştirilir ve çıktılar katılımcıları değerlendirmek için kullanılmaz
- Gözden geçirme teknikleri, hedeflere ulaşmanın uygun olacağı şekilde ve yazılımın çeşidine ve seviyesine ve ayrıca gözden geçiricilere uygun olacak şekilde uygulanır
- Hata tanımlama konusunda etkinliği artırmak için uygun olan durumlarda kontrol listeleri ve roller kullanılır
- Gözden geçirme tekniklerinde, özellikle de teftiş gibi daha resmi tekniklerde eğitim verilir
- Yönetim iyi bir gözden geçirme sürecini destekler (örn. proje zamanlamalarında gözden geçirme aktiviteleri için yeterli zamanı vererek)
- Öğrenme ve süreç iyileştirmesine vurgu yapılır

3.3 Araçlarla Gerçekleştirilen Statik Analiz (K2)	*20 dakika*

Terimler

Derleyici, karmaşıklık, kontrol akışı, veri akışı, statik analiz

Genel Bilgi

Statik analizin hedefi, yazılımın kaynak kodundaki ve yazılım modellerindeki hataları bulmaktır. Statik analiz, incelenen yazılım yürütülmeden gerçekleştirilir; dinamik test ise yazılım kodunu yürütür. Statik analiz, dinamik testte bulunması zor olan hataları bulabilir. Statik analiz araçları, program kodunu (örn. kontrol akışı ve veri akışı) ve HTML ile XML gibi oluşturulan çıktıyı analiz eder.

Statik analizin avantajları şunlardır:

- Testler yürütülmeden önce hataların erken tespiti
- Yüksek karmaşıklık ölçüsü gibi metriklerin hesaplanmasıyla koddaki veya tasarımdaki şüpheli durumlarla ilgili erken uyarı
- Dinamik test ile kolayca bulunamayan hataların belirlenmesi
- Yazılım modellerindeki bağımlılıkların ve tutarsızlıkların saptanması, linkler gibi
- İyileştirilmiş kod ve tasarım sürdürülebilirliği
- Uyarıların geliştirme sırasında dikkate alınması durumunda hataların önlenmesi

Statik analiz araçları tarafından bulunan genel hatalar şöyledir:

- Değer atanmamış değişkenin yanlışlıkla referans gösterildiği durumlar
- Modüller ve bileşenler arasında tutarsız arayüzler
- Kullanılmayan veya yanlış şekilde tanımlanmış değişkenler
- Ulaşılamayan, çağrılmayan (ölü) kod
- Eksik ve hatalı mantık (sonsuz döngüler)
- Çok karmaşık yapılar
- Programlama standardı ihlalleri
- Güvenlik açıkları
- Kod ve yazılım modellerinde söz dizimi ihlalleri

Statik analiz araçları genellikle bileşen ve entegrasyon testi öncesinde ve sırasında veya yapılandırma yönetimi araçlarına kod teslim ederken yazılımcılar tarafından (önceden belirlenen kurallara veya programlama standartlarına karşı kontrol etme) ya da yazılım modellemesi sırasında tasarımcılar tarafından kullanılır. Statik analiz araçları çok sayıda uyarı mesajı verebilir ve aracın en etkin şekilde kullanılması için bu uyarı mesajlarının iyi bir şekilde yönetilmesi gerekir.

Derleyiciler statik analize destek olabilir metriklerin hesaplanması gibi konular da dahil olmak üzere

Referanslar

3.2 IEEE 1028
3.2.2 Gilb, 1993, van Veenendaal, 2004
3.2.4 Gilb, 1993, IEEE 1028
3.3 van Veenendaal, 2004

4. Test Tasarım Teknikleri (K4)	285 dakika

Test Tasarım Teknikleri için Öğrenme Hedefleri

Hedefler, her bir modülü tamamladıktan sonra neler yapabileceğinizi tanımlar.

4.1 Test Geliştirme Süreci (K3)

ÖH-4.1.1 Test tasarımı, test senaryosu ve test prosedürü arasındaki farkı belirtme (K2)

ÖH-4.1.2 Test koşulu, test senaryosu ve test prosedürü terimlerini karşılaştırma (K2)

ÖH-4.1.3 Test senaryolarının kalitesini gereksinimlerin ve beklenen sonuçların izlenebilirliği bağlamında değerlendirme (K2)

ÖH-4.1.4 Test senaryolarını test prosedürlerine çevirme (K3)

4.2 Test Tasarım Tekniği Kategorileri (K2)

ÖH-4.2.1 Gereksinim bazlı (kara kutu) ve yapı bazlı (beyaz kutu) test tasarım tekniklerinin faydalarını hatırlama ve her biri için yaygın teknikleri listeleme (K1)

ÖH-4.2.2 Gereksinim bazlı test, yapı bazlı test ve tecrübeye dayalı test arasındaki özellikleri, ortak noktaları ve farkları açıklama (K2)

4.3 Gereksinim Bazlı veya Kara Kutu Teknikleri (K3)

ÖH-4.3.1 Denklik paylarına ayırma, sınır değer analizi, karar tabloları ve durum geçişi diyagramlarını/tablolarını kullanarak belirli yazılım modellerinden test senaryoları yazma (K3)

ÖH-4.3.2 Dört test tekniğinden her birinin amacını, hangi tekniğin hangi test seviyesi ve çeşidinde kullanabildiğini ve kapsamın nasıl ölçüldüğünü açıklama (K2)

ÖH-4.3.3 Kullanım senaryosu testi kavramını ve avantajlarını açıklama (K2)

4.4 Yapı Bazlı veya Beyaz Kutu Teknikleri (K4)

ÖH-4.4.1 Kod kapsamı kavramını ve değerini açıklama (K2)

ÖH-4.4.2 Komut ve karar kapsamı kavramlarını açıklama ve bu kavramların neden bileşen testi dışındaki test seviyelerinde (örn. sistem seviyesindeki iş prosedürlerinde) kullanılabildiğini açıklama (K2)

ÖH-4.4.3 Komut ve karar test tasarım tekniklerini kullanarak belirli kontrol akışlarından test senaryoları yazma (K3)

ÖH-4.4.4 Komut ve karar kapsamının belirlenen çıkış kriterine göre tamamlanıp tamamlanmadığını değerlendirme. (K4)

4.5 Tecrübeye Dayalı Teknikler (K2)

ÖH-4.5.1 Yaygın hatalarla ilgili sezgiye, tecrübeye ve bilgiye dayalı test senaryoları yazma nedenlerini hatırlama (K1)

ÖH-4.5.2 Tecrübeye dayalı tekniklerle spesifikasyon bazlı test tekniklerini karşılaştırma (K2)

4.6 Test Tekniklerini Seçme (K2)

ÖH-4.6.1 Belirli bir bağlam, test esası, model ve yazılım özelliklerine uygunluğuna göre test tasarım tekniklerini sınıflandırma (K2)

4.1 Test Geliştirme Süreci (K3) — *15 dakika*

Terimler

Test senaryosu, test tasarımı, test yürütme çizelgesi, test prosedürü, test komut dosyası, izlenebilirlik

Genel Bilgi

Bu bölümde anlatılan test geliştirme süreci, çok az sayıda dokümantasyon veya hiç doküman olmadan gayri resmi süreçten, çok resmi sürece kadar çeşitli şekillerde gerçekleştirilebilir. Resmiyet seviyesi testin bağlamına göre değişir, bunlar arasında testin ve geliştirme süreçlerinin olgunluğu, zaman kısıtlamaları, emniyet veya mevzuat ve katılan kişiler yer alabilir.

Test analizi sırasında test esası test edilecek şeye karar vermek için, örneğin test koşullarını tanımlamak için analiz edilir. Test koşulu bir veya daha fazla test senaryosuyla doğrulanabilen bir öğe veya olaydır (işlev, işlem, kalite karakteristiği veya yapısal öğe).

Test koşullarından, gereksinimlere uzanacak şekilde bir geri izlenebilirlik oluşturmak, gereksinimler değiştiğinde etkin bir etki analizi gerçekleştirilmesine olanak tanır ve bir dizi teste yönelik gereksinim kapsamının belirlenmesini sağlar. Test analizi sırasında, risklere dayanarak kullanılacak test tasarım tekniklerini seçmek için test yaklaşımı belirlenir (risk analizi hakkında daha fazla bilgi için bkz. Bölüm 5).

Test tasarımı sırasında test senaryoları ve test verisi oluşturulur ve belirlenir. Test senaryosu, belirli test hedeflerini veya test koşullarını kapsayacak şekilde belirlenmiş bir dizi girdi değeri, yürütme önkoşulu, beklenen sonuç ve yürütme artkoşulu içerir. "Yazılım Testi Dokümantasyonu Standardı" (IEEE STD 829-1998), test tasarımlarını (test koşullarını içeren) ve test senaryolarının içeriğini tanımlar.

Beklenen sonuçlar, bir test senaryosunun parçası olarak üretilir ve çıktıları, veri ve durumlardaki değişiklikleri ve testin diğer sonuçlarını içerir. Beklenen sonuçlar tanımlanmamışsa uygun gözüken fakat hatalı olan bir sonuç doğru sonuç olarak yorumlanabilir. İdeal olarak beklenen sonuçlar test yürütmeden önce tanımlanmalıdır.

Test uyarlama sırasında test senaryoları ve test prosedürü geliştirilir, uyarlanır, önceliklendirilir ve organize edilir (IEEE STD 829-1998). Test prosedürü testin yürütüleceği işlem sırasını belirler. Testler, test yürütme aracı kullanılarak yürütülüyorsa işlemlerin sıralaması test komut dosyasında belirlenir (buna otomatik test prosedürü denir).

Çeşitli test prosedürleri ve test komut dosyaları test yürütme çizelgesine dönüşür. Bu çizelge çeşitli test prosedürlerinin ve test komut dosyalarının hangi sırayla yürütüleceğini belirler. Test yürütme çizelgesi, regresyon, önceliklendirme, teknik ve mantıksal bağımlılıklar gibi faktörleri göz önünde bulundurur.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

4.2 Test Tasarım Tekniği Kategorileri (K2)	*15 dakika*

Terimler

Kara kutu test tasarım tekniği, tecrübeye dayalı test tasarım tekniği, test tasarım tekniği, beyaz kutu test tasarım tekniği

Genel Bilgi

Test tasarım tekniğinin amacı test koşullarını, test senaryolarını ve test verisini tanımlamaktır.

Test tekniklerinin kara kutu veya beyaz kutu olarak tanımlanması klasik bir ayırma yoludur. Kara kutu test tasarım teknikleri (spesifikasyon bazlı teknikler adı da verilir); test koşullarını, test senaryolarını veya test verisini türetmek için test esası dokümanlarının analizine dayanan bir yoldur. Fonksiyonel ve fonksiyonel olmayan testleri içerir. Kara kutu testi, test edilecek bileşenin veya sistemin dahili yapısı ile ilgili hiçbir bilgiyi kullanmaz. Beyaz kutu test tasarım teknikleri (yapısal veya yapı bazlı teknikler adı da verilir), bileşen veya sistem iç yapısının analizine dayanır. Test edilmesi gereken şeye karar vermeleri için yazılımcıların, test uzmanlarının ve kullanıcıların tecrübelerini kullanmak amacıyla kara kutu ve beyaz kutu testi, tecrübeye dayalı tekniklerle de bir araya getirilebilir.

Bazı teknikler net bir şekilde tek kategoriye sahip olurken diğerleri birden fazla kategori öğesine sahip olabilir.

Bu ders programı gereksinim bazlı test tasarım tekniklerini kara kutu teknikleri ve yapı bazlı test tasarım tekniklerini beyaz kutu teknikleri olarak ele almaktadır. Ayrıca tecrübeye dayalı test tasarım teknikleri de kapsam içindedir.

Gereksinim bazlı test tasarım tekniklerinin bilinen özellikleri şöyledir:

- Modeller, çözülecek problemin gereksinimlerini belirlemek için kullanılır
- Test senaryoları bu modellerden sistematik olarak türetilebilir

Yapı bazlı test tasarım tekniklerinin bilinen özellikleri şöyledir:

- Yazılımın iç çalışma mantığı ile ilgili bilgiler test senaryoları türetmek için kullanılır (kod ve detaylı tasarım bilgileri)
- Mevcut test senaryolarıyla yakalanan kapsam derecesi ölçülerek kapsamı artırmak için daha fazla test senaryosu yazılabilir

Tecrübeye dayalı test tasarım tekniklerinin bilinen özellikleri şöyledir:

- Test senaryoları türetmek için kişilerin bilgisi ve tecrübesi kullanılır
- Yazılım, yazılımın kullanımı ve ortamı hakkında test uzmanlarının, yazılımcıların, kullanıcıların ve diğer paydaşların bilgi birikimi, bilgi kaynaklarından biridir
- Olası hatalar ve bu hataların dağılımı da diğer bir bilgi kaynağıdır

4.3 Spesifikasyon Bazlı veya Kara Kutu Teknikleri (K3)	*150 dakika*

Terimler

Sınır değer analizi, karar tablosu testi, denklik paylarına ayırma, durum geçişi testi, kullanım senaryosu testi

4.3.1 Denklik Paylarına Ayırma (K3)

Denklik paylarına ayırmada yazılımın girdileri aynı davranışı göstermesi beklenen gruplara ayrılır, böylece aynı şekilde ele alınabilirler. Denklik payları (veya sınıflar), hem geçerli veriler (kabul edilmesi gereken değerler) hem de geçersiz veriler (reddedilmesi gereken değerler) için oluşturulabilir. Paylar aynı zamanda çıktılar, dahili değerler, zamanla ilgili değerler (bir olaydan önce veya sonra) ve arayüz parametreleri (entegrasyon testi sırasında test edilen entegre edilmiş bileşenler) için de tanımlanabilir. Testler tüm geçerli ve geçersiz payları kapsayacak şekilde tasarlanabilir. Denklik paylarına ayırma tüm test seviyelerinde uygulanabilir.

Denklik paylarına ayırma, girdi ve çıktı kapsam hedeflerine ulaşmak için kullanılabilir. Manuel girdilere, bir yazılıma arayüzler aracılığı ile giriş yapılmasına veya entegrasyon testindeki arayüz parametrelerinde de uygulanabilir.

4.3.2 Sınır Değer Analizi (K3)

Denklik paylarının uç noktalarındaki girdilerin hataya sebep olma olasığı daha yüksek olduğu için bu alanların daha yoğunlukla test edilmesi gerekmektedir. Bu teknik kullanılarak bu sınır değerlerinin bulunması amaçlanmaktadır. Bir payın maksimum ve minimum değerleri, sınır değerleridir. Geçerli bir payın sınır değeri, geçerli sınır değeridir; geçersiz bir payın sınırı ise geçersiz sınır değeridir. Testler geçerli ve geçersiz sınır değerlerini kapsayacak şekilde tasarlanabilir. Test senaryoları tasarlanırken her bir sınır değeri için bir test seçilir.

Sınır değer analizi tüm test seviyelerinde uygulanabilir. Uygulaması kolaydır ve hata bulma becerisi yüksektir. Ayrıntılı gereksinimler ilginç sınırları belirlemeye yardımcı olur.

Bu teknik genellikle denklik paylarına ayırma veya diğer kara kutu test tasarım tekniklerinin bir uzantısı olarak düşünülür. Ekranda kullanıcı girdisi için denklik sınıfında veya zaman aralıklarında (zaman aşımı, işlem hızı gereksinimleri) veya tablo aralıklarında (örn. tablo boyutu 256*256) kullanılabilir.

4.3.3 Karar Tablosu Testi (K3)

Mantıksal koşulları içeren gereksinimleri yakalamak ve yazılım iç tasarım mantığını dokümante etmek için karar tabloları iyi bir tekniktir. Bir yazılımdaki karmaşık iş kurallarını kaydetmek için de kullanılabilirler. Karar tabloları oluştururken gereksinimler analiz edilir ve yazılımın koşulları ve eylemleri belirlenir. Girdi koşulları ve eylemler sıklıkla doğru veya yanlış (Boolean) olacakları şekilde ifade edilir. Karar tablosu testi, tetikleme koşullarını, genellikle tüm girdi koşulları için doğru ve yanlış kombinasyonlarını ve her bir koşul kombinasyonu için ortaya çıkan eylemleri, sonuçları içerir. Tablonun her bir sütunu, farklı bir koşul kombinasyonunu tanımlayan bir iş kuralına karşılık gelir ve bu kuralla bağlantılı eylemlerin yürütülmesi ile sonuçlanır. Karar tablosu testinde yaygın şekilde kullanılan test kapsamında amaç tabloda yer alan sütunlardan her biri çin en az bir testin yürütülmesi şeklindedir.

Karar tablosu testinin güçlü yanı, test sırasında gözden kaçabilecek koşul kombinasyonlarının net bir şekilde listelenmesidir. Yazılım davranışının birden fazla mantıksal karara bağlı olduğu tüm durumlarda uygulanabilir.

4.3.4 Durum Geçişi Testi (K3)

Yazılımın davranışı mevcut veya geçmişteki durumuna göre değişiklik gösterebilir. Bu tür davranışlar sergileyen yazılımlar bir durum geçiş diyagramı ile gösterilebilir. Durum geçiş diyagramları test uzmanının yazılımın alabileceği durumları, durumlar arasındaki geçişleri, durum değişikliklerini (geçişleri) tetikleyen girdileri veya olayları ve bu geçişler sonucunda oluşabilecek eylemleri görüntülemesini sağlar. Test edilen sistemin veya nesnenin durumları ayrı ayrıdır, tanımlanabilir ve ölçülebilir sayıdadır.

Durum tablosu, durumlar ve girdiler arasındaki ilişkiyi gösterir ve geçersiz olan olası geçişleri ortaya çıkarabilir.

Durumların genel sıralamasını kapsayacak, her durumu kapsayacak, her geçişi deneyecek, belirlenmiş geçiş sıralamalarını deneyecek veya geçersiz geçişleri test edecek testler tasarlanabilir.

Durum geçişi testi genel olarak en fazla gömülü yazılımlarda ve teknik otomasyonda kullanılır. Ayrıca bu teknikle nesnelerin modellenmesi veya ekran-diyalog akışının test edilmesi (örn. İnternet uygulamaları veya iş senaryoları için) mümkündür.

4.3.5 Kullanım Senaryosu Testi (K2)

Test uzmanları kullanım senaryolarını kullanarak testler türetilebilir. Kullanım senaryosu, aktörler ve sistem arasındaki etkileşimleri tanımlayarak; bu etkileşimler sonucunda üretilen değeri gösterir. Kullanım senaryoları soyut seviyede (iş kullanım senaryosu, teknolojiden bağımsız, iş süreç seviyesi) veya sistem seviyesinde (sistem fonksiyonalite seviyesinde sistem kullanım senaryosu) tanımlanabilir. Her bir kullanım senaryosunda, kullanım senaryosunun başarılı bir şekilde çalışması için karşılanması gereken önkoşullar bulunur. Her bir kullanım senaryosu, kullanım senaryosu tamamlandıktan sonra gözlemlenebilir sonuçları ve sistemin son durumunu içeren artkoşullarla sona erer. Kullanım senaryosunda genellikle bir ana (en olası) senaryo ve alternatif senaryolar bulunur.

Kullanım senaryoları, gerçek kullanımlara dayanarak sistem boyunca "süreç" akışını tanımlar, bu nedenle kullanım senaryolarından türetilen test senaryoları, sistemin gerçek dünyada kullanımı sırasında süreç akışlarında hataları ortaya çıkarmanın en kolay yoludur. Kullanım senaryoları, müşteri/kullanıcı katılımı ile kabul testleri tasarlamada da çok kullanışlıdır. Ayrıca, bağımsız bileşen testlerinin göremeyeceği şekilde, farklı bileşenlerin etkileşiminin neden olduğu entegrasyon hatalarının gün yüzüne çıkarılmasını sağlar. Kullanım senaryolarından test senaryoları tasarlamak, diğer gereksinim bazlı test teknikleri ile de bir araya getirilebilir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

4.4 Yapı Bazlı veya Beyaz Kutu Teknikleri (K4)	*60 dakika*

Terimler

Kod kapsamı, karar kapsamı, komut kapsama yüzdesi, yapı bazlı testler

Genel Bilgi

Yapı bazlı veya beyaz kutu testi, aşağıdaki örneklerde görülebileceği üzere yazılımın iç çalışma mantığına dayanır:

- Bileşen seviyesi: bir yazılım bileşeninin yapısı, örn. komutlar, kararlar, dallar ve yollar
- Entegrasyon seviyesi: Test için ele alınan yapı bileşenlerin birbirlerini nasıl çağırdığını gösteren bir çağrı ağacı olabilir (örnek modüllerin diğer modülleri çağırdığı bir diyagram)
- Sistem seviyesi: Yapı; menü yapısı, iş süreci veya web sayfası yapısı olabilir

Bu bölümde kod yapısıyla ilgili üç çeşit test tekniğinden bahsedilecektir: komut, karar ve dal test teknikleri.

4.4.1 Komut Testi ve Kapsam (K4)

Komut testinde kapsam çalıştırılan komutların yüzdesi ele alınarak belirlenir. Komut kapsamı yüzdesi, test senaryoları tarafından kapsanan (tasarlanmış veya yürütülmüş) yürütülebilir komut sayısının test edilen koddaki tüm yürütülebilir komutların sayısına bölünmesiyle elde edilir.

4.4.2 Karar Testi ve Kapsam (K4)

Dal testi ile ilgili olan karar kapsamı, bir test senaryo grubu tarafından oluşturulmuş karar çıktılarını (örn. bir IF komutunun doğru ve yanlış sonuç üreten seçenekleri) dikkate alır

Karar kapsamı, test edilen kodda test grubu tarafından oluşturulmuş karar çıktılarının koddaki tüm karar çıktılarına bölünmesiyle elde edilir.

Karar testinde karar noktaları boyunca bir kontrol akışını izlendiği için karar testi bir çeşit kontrol akış testi biçimidir. Karar kapsamı, komut kapsamından daha güçlüdür; %100 karar kapsamı %100 komut kapsamını sağlar, ancak bunun tersi olmaz.

4.4.3 Diğer Yapı Bazlı Teknikler (K1)

Karar kapsamının ötesinde daha güçlü yapısal kapsam seviyeleri vardır; örn. koşul kapsamı ve çoklu koşul kapsamı.

Kapsam kavramı diğer test seviyelerinde de uygulanabilir. Örneğin, entegrasyon seviyesinde, bir test senaryo grubu tarafından denenmiş modüllerin, bileşenlerin veya sınıfların yüzdesi, modül, bileşen veya sınıf kapsamı olarak ifade edilebilir.

Kodun yapısal testi için araç desteği oldukça yararlıdır.

4.5 Tecrübeye Dayalı Teknikler (K2) — *30 dakika*

Terimler

Keşif testi, (kusur ortaya çıkarmaya yönelik) saldırı

Genel Bilgi

Tecrübeye dayalı testte, testler benzer uygulamalar ve teknolojilerle daha önce çalışmış test uzmanının becerisine, sezgilerine ve tecrübesine dayanılarak türetilir. Sistematik teknikleri artırmak için kullanıldığında bu teknikler, resmi teknikler tarafından kolayca yakalanamayan özel testleri belirlemek için kullanılabilir. Fakat bu tekniğin etkisi test uzmanının tecrübesine bağlı olarak çeşitlilik gösterecektir.

Yaygın olarak kullanılan tecrübeye dayalı tekniklerden birisi de hata tahminlemedir. Genellikle test uzmanları tecrübelerine dayanarak hataları tahmin eder. Hata tahminleme tekniğine yönelik bir yaklaşım, olası hataların listesini çıkarmak ve bu hataları hedef alan (onlara saldıran) testler tasarlamaktır. Bu sistematik yaklaşıma kusur ortaya çıkarmaya yönelik saldırı adı verilir. Hata listeleri, tecrübeye, mevcut hata ve arıza verilerine ve yazılımın neden başarısız olduğu ile ilgili yaygın bilgilere dayanarak oluşturulur.

Keşif testi, test hedeflerini içeren bir test başlatma dokümanına dayanan ve belli zaman aralıkları içinde gerçekleştirilen test tasarımının, test yürütmenin, test kaydı tutmanın ve öğrenmenin eş zamanlı olarak yapıldığı bir test çeşididir. Bu yaklaşım, az veya yetersiz gereksinimler ve ciddi zaman sınırlaması olan durumlarda ya da daha resmi testleri artırmak ya da tamamlamak için oldukça kullanışlıdır. Test sürecinde kontrol olarak kullanılabilir ve en ciddi hataların bulunmasını sağlar.

4.6 Test Tekniklerini Seçme (K2) — *15 dakika*

Terimler
Spesifik terim yoktur.

Genel Bilgi
Kullanılacak test tekniklerini seçmek birçok faktöre dayanır: sistem çeşidi, mevzuat, müşteri veya sözleşme gereksinimleri, risk seviyesi, risk çeşidi, test hedefi, mevcut dokümanlar, test uzmanlarının bilgi birikimi, süre ve bütçe, yazılım geliştirme yaşam döngüsü, kullanım senaryosu modelleri ve bulunan hata türleri konusunda önceki tecrübeler.

Bazı teknikler belirli durumlara ve test seviyelerine daha uygunken diğerleri tüm test seviyelerinde uygulanabilir.

Test senaryoları oluştururken test uzmanları genellikle, test edilen nesnenin yeterli şekilde kapsanmasını sağlamak için süreci, kuralı ve veri güdümlü teknikleri içeren bir test tekniği kombinasyonu kullanır.

Referanslar

4.1 Craig, 2002, Hetzel, 1988, IEEE STD 829-1998
4.2 Beizer, 1990, Copeland, 2004
4.3.1 Copeland, 2004, Myers, 1979
4.3.2 Copeland, 2004, Myers, 1979
4.3.3 Beizer, 1990, Copeland, 2004
4.3.4 Beizer, 1990, Copeland, 2004
4.3.5 Copeland, 2004
4.4.3 Beizer, 1990, Copeland, 2004
4.5 Kaner, 2002
4.6 Beizer, 1990, Copeland, 2004

5. Test Yönetimi (K3)	170 dakika

Test Yönetimi için Öğrenme Hedefleri

Hedefler, her bir modülü tamamladıktan sonra neler yapabileceğinizi tanımlar.

5.1 Test Organizasyonu (K2)

ÖH-5.1.1 Bağımsız testin önemini kavrama (K1)
ÖH-5.1.2 Organizasyon içinde bağımsız testin avantajlarını ve sakıncalarını açıklama (K2)
ÖH-5.1.3 Test ekibinin oluşturulması için düşünülecek farklı ekip üyelerini hatırlama (K1)
ÖH5.1.4 Tipik bir test uzmanının ve test liderinin görevlerini hatırlama (K1)

5.2 Test Planlama ve Tahminleme (K3)

ÖH-5.2.1 Test planlamanın farklı seviyelerini ve hedeflerini hatırlama (K1)
ÖH-5.2.2 "Yazılım Testi Dokümantasyonu Standardı"na (IEEE Std 829-1998) göre test planı, test tasarım spesifikasyonu ve test prosedürü belgelerinin amacını ve içeriğini özetleme (K2)
ÖH-5.2.3 Analitik, model bazlı, metotlu, süreç/standart uyumlu, dinamik/bulgusal, istişari ve regresyon hassasiyetli gibi kavramsal olarak farklı test yaklaşımları arasındaki farkı belirtme (K2)
ÖH-5.2.4 Test planlama ve test yürütme arasındaki farkı belirtme (K2)
ÖH-5.2.5 Belirli bir dizi test senaryosu için, önceliklendirmeyi, teknik ve mantıksal bağımlılıkları göz önüne alarak test yürütme çizelgesi yazma (K3)
ÖH-5.2.6 Test planlama sırasında düşünülmesi gereken test hazırlama ve yürütme işlemlerini listeleme (K1)
ÖH-5.2.7 Test etme ile ilgili çabaları etkileyen genel faktörleri hatırlama (K1)
ÖH-5.2.8 Kavramsal olarak farklı iki tahminleme yaklaşımı arasındaki farkı belirtme: metrik bazlı yaklaşım ve uzman bazlı yaklaşım (K2)
ÖH-5.2.9 Spesifik test seviyeleri ve test senaryosu grupları için yeterli giriş ve çıkış kriterini hatırlama/doğrulama (örn. entegrasyon testi, kabul testi veya kullanılabilirlik testine yönelik test senaryoları için) (K2)

5.3 Test İlerleme Gözetimi ve Kontrolü (K2)

ÖH-5.3.1 Test hazırlığını ve yürütmeyi monitörlemek için kullanılan genel metrikleri hatırlama (K1)
ÖH-5.3.2 Amaç ve kullanım ile ilgili test raporlama ve test kontrol (örn. bulunan ve düzeltilen hatalar, başarılı ve başarısız olan testler) için test metriklerini açıklama ve karşılaştırma (K2)
ÖH-5.3.3 "Yazılım Testi Dokümantasyonu Standardı"na (IEEE Std 829-1998) göre test özet raporu dokümanının amacını ve içeriğini özetleme (K2)

5.4 Yapılandırma Yönetimi (K2)

ÖH-5.4.1 Yapılandırma yönetiminin test etmeyi nasıl desteklediğini özetleme (K2)

5.5 Risk ve Test Etme (K2)

ÖH-5.5.1 Bir veya daha fazla paydaşın proje hedeflerine ulaşmasını engelleyecek olası bir problem olarak riski tanımlama (K2)
ÖH-5.5.2 Risk seviyesinin olasılığa (olma olasılığı) ve etkiye (olursa oluşturacağı zarar) göre belirlendiğini hatırlama (K1)
ÖH-5.5.3 Proje ve ürün riskleri arasındaki farkı belirtme (K2)
ÖH-5.5.4 Genel ürün ve proje risklerini hatırlama (K1)
ÖH-5.5.5 Test planlama için risk analizinin ve risk yönetiminin nasıl kullanılabileceğini örnekler vererek açıklama (K2)

5.6 Olay Yönetimi (K3)

ÖH-5.6.1 "Yazılım Testi Dokümantasyonu Standardı"na (IEEE Std 829-1998) göre olay raporunun içeriğini hatırlama (K1)

ÖH-5.6.2 Test sırasında bir arızanın gözlemlenmesini kapsayan bir olay raporu yazma. (K3)

5.1 Test Organizasyonu (K2) — *30 dakika*

Terimler

Test uzmanı, test lideri, test yöneticisi

5.1.1 Test Organizasyonu ve Bağımsızlık (K2)

Testte ve gözden geçirmelerde hataları bulmanın etkinliği, bağımsız test uzmanları kullanılarak iyileştirilebilir. Bağımsızlık seviyeleri aşağıdaki şekilde listelenebilir:

- Bağımsız test uzmanı yoktur; yazılımcılar kendi kodlarını test eder
- Yazılım geliştirme ekiplerinin içindeki bağımsız test uzmanları
- Proje yönetimine veya yönetici kadroya rapor veren, organizasyon içindeki bağımsız test ekibi
- Şirket dışından bağımsız test uzmanları
- Kullanılabilirlik testi uzmanları, güvenlik testi uzmanları veya sertifikasyon testi uzmanları (bir yazılım ürününü standartlara ve düzenlemelere göre sertifikalandıran) gibi spesifik test çeşitleri için bağımsız test uzmanları
- Dış kaynak kullanımı hizmeti veren şirketlerden gelen bağımsız test uzmanları

Büyük, karmaşık ve riskli olan projelerde, birkaç seviyenin veya tüm seviyelerin bağımsız test uzmanları tarafından gerçekleştirilmesi genellikle en iyi uygulamadır. Yazılım geliştirme ekibi de özellikle daha alt seviyelerde teste katılabilir, ancak yeterince objektif olmadıklarından etkinlikleri sınırlıdır. Bağımsız test uzmanları, sadece testleri yürütmenin yanında test süreçlerinin ve kurallarının gereksinimlerini belirleme yetkisine ve sorumluluğuna da sahip olabilir. Test uzmanları bu tür süreç iyileştirmesiyle ilgili sorumlulukları ancak müşteri yönetimi tarafından kendilerine yeterince yetki, insiyatif ve net bir çerçeve çizildiği zaman almalıdır.

Bağımsızlığın avantajları arasında şunlar yer alır:

- Bağımsız test uzmanları farklı bakış açılarıyla farklı hataları görür ve önyargısızdır
- Bağımsız bir test uzmanı, yazılımın analizi ve uyarlanması sırasında yapılmış olan varsayımları doğrulayabilir

Bağımsızlığın dezavantajları arasında şunlar yer alır:

- Yazılım geliştirme ekibinden soyutlanma (tamamen bağımsız bir test modeli uygulanırsa)
- Yazılım geliştiricilerin kalite konusundaki sorumluluk hissini kaybetmesi
- Bağımsız test uzmanları şirket bünyesinde bir darboğaz olarak görülebilir veya yazılımın piyasaya çıkmasının gecikmesi konusunda suçlanabilir

Test görevleri test uzmanları tarafından yapılabileceği gibi proje yöneticisi, kalite yöneticisi, yazılımcı, alan uzmanı, altyapı veya BT operatörleri gibi diğer rollere sahip biri tarafından da yapılabilir.

5.1.2 Test Liderinin ve Test Uzmanının Görevleri (K1)

Bu ders programında iki test pozisyonundan bahsedilmektedir; test lideri ve test uzmanı. Bu iki role sahip kişiler tarafından yapılan işlemler ve görevler, projenin ve ürünün bağlamına, o rolü üstlenen kişilere ve kuruluşa göre değişir.

Bazı zamanlarda test liderine test yöneticisi veya test koordinatörü adı verilir. Test liderinin rolü, proje yöneticisi, yazılım geliştirme yöneticisi, kalite güvence yöneticisi veya bir test grubunun yöneticisi tarafından gerçekleştirilebilir. Daha büyük projelerde test lideri rolü ikiye ayrılabilir: test lideri ve test yöneticisi. Kısım 1.4'te açıklandığı gibi genellikle test lideri test aktivitelerini ve görevlerini planlar, izler ve kontrol eder.

Genel test lideri görevleri şunları içerebilir:

- Test stratejisini koordine etme, proje yöneticileri ve diğer paydaşlarla planlama
- Projenin test stratejisini ve kuruluşun test politikasını yazma veya gözden geçirme

- Test yaklaşımının diğer proje adımlarını etkilemesini sağlama örneğin entegrasyon planlama
- Test hedeflerini ve riskleri göz önünde bulundurarak testleri planlama; test yaklaşımlarını seçme, testin süresini, eforu ve maliyeti hesaplama, kaynakları hazır etme, test seviyelerini ve döngüleri belirleme, olay yönetimini planlama
- Testlerin analizini, hazırlığını, uyarlanmasını ve yürütülmesini başlatma, test sonuçlarını monitörleme ve çıkış kriterlerini kontrol etme
- Test sonuçlarına ve ilerlemeye (bazen durum raporlarında belgelenir) dayanarak planlama üzerinde değişikliiklere gitme ve sorunları telafi etmek için gerekli eylemleri gerçekleştirme
- Test yazılımlarının izlenebilirliği için ile yapılandırma yönetimini hazırlama
- Test ilerleyişini ölçmek, testin ve yazılımın kalitesini değerlendirmek için uygun metrikleri belirleme
- Hangi test senaryolarının hangi dereceye kadar ve nasıl otomasyona geçirileceğine karar verme
- Testi destekleyecek araçları seçme ve test uzmanlarının araç kullanımı konusunda eğitimlere katılmasını sağlama
- Test ortamının uyarlanması ile ilgili karar verme
- Test sırasında toplanan bilgilere dayanarak test özet raporları yazma

Genel test uzmanının görevleri şunları içerebilir:

- Test planlarını gözden geçirme ve bunlara katkı sağlama
- Kullanıcı gereksinimlerini ve test için oluşturulmuş modelleri analiz etme, gözden geçirme ve değerlendirme
- Test gereksinimlerini oluşturma
- Test ortamını hazırlama (genellikle sistem yöneticisi ve ağ yönetimi ile koordineli bir şekilde)
- Test verisini hazırlama ve alma
- Tüm test seviyelerinde testleri uyarlama, testleri yürütme ve kayıt altına alma, sonuçları değerlendirme ve beklenen sonuçlardan sapmaları belgeleme
- Gerektiğinde test yönetimini veya yönetim araçlarını ve test gözetimi (izleme) araçlarını kullanma
- Testleri otomasyona geçirme (bir yazılım geliştirici veya test otomasyonu uzmanından destek alınabilir)
- Bileşenlerin ve sistemlerin performansını ölçme (uygunsa)
- Diğerleri tarafından geliştirilen testleri gözden geçirme

Test analizi, test tasarımı, test çeşitleri veya test otomasyonu üzerinde çalışan kişiler bu rollerde uzman olabilir. Bulunulan test seviyesine, yazılım ve proje ile ilgili risklere bağlı olarak farklı kişiler test uzmanı rolünü üstlenebilir (belirli bir derecede bağımsız olarak). Genellikle bileşen ve entegrasyon seviyesindeki test uzmanları yazılım geliştiricilerdir. Kabul testi seviyesindeki testi yapan kişiler ise iş analistleri ve kullanıcılar olup operasyonel kabul testi gerçekleştirenler ise operatörlerdir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

5.2 Test Planlama ve Tahminleme (K3)	*40 dakika*

Terimler

Test yaklaşımı, test stratejisi

5.2.1 Test Planlama (K2)

Bu kısımda yazılım geliştirme ve uyarlama projeleri dahilinde ve bakım işlemleri için test planlamanın amacı açıklanmaktadır. Planlama bir master test planında yer alabileceği gibi sistem testi ve kabul testi gibi test seviyeleri için ayrı test planlarında da yer alabilir. Test planının ana hatlarına "Yazılım Testi Dokümantasyonu Standardı" (IEEE Std 829-1998) kapsamında değinilmektedir.

Organizasyonun test politikası, test kapsamı, hedefler, riskler, sınırlandırmalar, önem, test edilebilirlik ve kaynakların elverişliliği gibi faktörler planlamayı etkiler. Proje ve test ilerledikçe daha fazla bilgi ortaya çıkar ve test planına daha fazla ayrıntı eklenebilir.

Test planlama sürekli devam eden bir aktivitedir ve tüm yazılım yaşam döngüsü süreçlerinde ve aktivitelerinde ele alınır. Planın güncellenmesi için değişen riskleri tanımak adına test aktivitelerinden gelen geri bildirimler kullanılır.

5.2.2 Test Planlama Adımları (K3)

Yazılımın tamamı veya bir kısmına yönelik test planlama işlemleri şunları içerebilir:

- Kapsamı ve riskleri tanımlama ve testin hedeflerini belirleme
- Test seviyelerinin, giriş ve çıkış kriterinin tanımı da dahil testin genel yaklaşımını tanımlama
- Test aktivitelerini yazılım yaşam döngüsü adımlarıyla (alma, sağlama, geliştirme, operasyon ve bakım) entegre etme ve koordine etme
- Neyin test edileceği, hangi rollerin test işlemlerini uygulayacağı, test işlemlerinin nasıl yapılması gerektiği ve test sonuçlarının nasıl değerlendirilmesi gerektiği ile ilgili kararlar verme
- Test analizi ve tasarım aktivitelerinin zaman planlamasını yapma
- Test uyarlama, yürütme ve değerlendirmenin zaman planlamasını yapma
- Tanımlanan aktiviteler için kaynakları atama
- Test dokümantasyonu için miktarı, ayrıntı seviyesini, yapıyı ve şablonları tanımlama
- Test hazırlığı ve yürütme, hata çözümleme ve risk konularını monitörlemek ve kontrol etmek için metrikleri seçme
- Yeniden üretilebilir test hazırlığını ve yürütmeyi sağlamak amacıyla test prosedürlerinin ayrıntı seviyesini belirleme

5.2.3 Giriş Kriteri (K2)

Giriş kriteri teste ne zaman başlanacağını belirler (test seviyesinin başlangıcında veya bir dizi test yürütmeye hazır olduğunda).

Genellikle giriş kriteri aşağıdakileri kapsayabilir:

- Test ortamı elverişliliği ve hazırlık
- Test ortamında test aracı hazırlığı
- Test edilebilir kod elverişliliği
- Test verisi elverişliliği

5.2.4 Çıkış Kriteri (K2)

Çıkış kriteri testin ne zaman durdurulacağını belirler (test seviyesinin sonunda veya bir dizi test hedefine ulaşıldığında).

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Genellikle çıkış kriteri aşağıdakileri kapsayabilir:

- Kodun kapsamı, fonksiyonalite veya risk gibi bütünlük ölçümleri
- Hata yoğunluğu veya güvenilirlik ölçülerinin tahminleri
- Maliyet
- Düzeltilmeyen hatalar veya belirli alanlarda test kapsamının yeterli olmaması gibi riskler
- Piyasaya sunma tarihi gibi zaman planlamaları

5.2.5 Test Tahminlemesi (K2)

Test tahminlemesi için iki yaklaşım bulunmaktadır:

- Metrik bazlı yaklaşım: daha önceki veya benzer projelere ya da genel değerlere dayanarak test çabasını tahmin etme
- Uzman bazlı yaklaşım: testte yapılacak işlerin sahibi veya uzmanlar tarafından yapılan tahminlere dayanarak görevleri tahmin etme

Test eforu tahmin edildiğinde kaynaklar belirlenebilir ve bir zaman çizelgesi çizilebilir.

Test eforu birçok faktöre bağlı olabilir:

- Yazılımın özellikleri: test modelleri için kullanılan gereksinim ve diğer bilgilerin kalitesi (örn. test esası), yazılımın boyutu, problemli alanın karmaşıklığı, güvenilirlik ve güvenlik için gereksinimler ve dokümantasyon gereksinimleri
- Geliştirme sürecinin özellikleri: kuruluşun kararlılığı, kullanılan araçlar, test süreci, katılan kişilerin becerileri ve zaman kısıtlaması
- Test ürünü/çıktısı: hataların sayısı ve gereken yeniden çalışma eforu

5.2.6 Test Stratejisi, Test Yaklaşımı (K2)

Test yaklaşımı, bir test projesi için test stratejisinin uyarlanmasıdır. Test yaklaşımı, test planlarında ve test tasarımlarında tanımlanır ve düzenlenir. Genellikle test projesinin amacına ve risk değerlendirmesine dayanarak verilen kararları içerir. Test yaklaşımı test sürecini planlama, uygulanacak test tasarım tekniklerini ve test çeşitlerini seçme, giriş ve çıkış kriterini belirlemek için referans noktasıdır.

Seçilen yaklaşım bağlama göre değişir ve riskleri, tehlike ve emniyeti, elverişli kaynakları ve becerileri, teknolojiyi, sistemin doğasını (örn. paket yazılımlar), test hedefleri ve mevzuatı göz önünde bulundurabilir.

Test yaklaşımlarına aşağıdakiler örnek verilebilir:

- Analitik yaklaşımlar: testin en riskli alanlara yönlendirildiği risk bazlı test gibi
- Model bazlı yaklaşımlar: Arıza (güvenilirlik büyüme modelleri gibi) veya kullanım oranları gibi (operasyonel profiller gibi) istatistiksel bilgileri kullanan stokastik testler
- Metotlu yaklaşımlar: Arıza bazlı (hata tahminleme ve kusur ortaya çıkarmaya yönelik saldırıları içeren), tecrübeye dayalı, kontrol listesi bazlı ve kalite özelliği bazlı gibi
- Süreç veya standartlara uyumlu yaklaşımlar: Endüstriye özel standartlar tarafından belirlenenler veya çeşitli çevik metotlar gibi
- Dinamik ve doğaçlama yaklaşımlar: Testin önceden planlanmadığı, yazılımın verdiği tepkiler ve bulunan hatalar doğrultusunda test tasarımının, yürütmenin ve değerlendirmenin aynı anda yapıldığı test yaklaşımları, keşif testleri gibi
- İstişari yaklaşımlar: Test kapsamının teknoloji ve/veya test ekibi dışındaki alan uzmanlarının önerileri ve rehberlikleri ile belirlendiği yaklaşımlar
- Regresyon hassasiyetli yaklaşımlar: Var olan test materyalinin, test komut dosyalarının ve test gruplarının yeniden kullanımını içeren yaklaşımlar gibi

Farklı yaklaşımlar birleştirilebilir, örneğin risk bazlı dinamik yaklaşım.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

5.3 Test İlerleme Gözetimi ve Kontrolü (K2)	*20 dakika*

Terimler

Hata yoğunluğu, arıza oranı, test kontrol, test gözetimi, test özet raporu

5.3.1 Test İlerleme Gözetimi (K1)

Test gözetiminin amacı test işlemleri hakkında geri bildirim ve şeffaflık sağlamaktır. Takip edilecek bilgiler manuel veya otomatik olarak toplanabilir. Bu bilgiler çıkış kriterini ölçmek için kullanılabilir, örneğin kapsam. Planlanan zaman çizelgesine ve bütçeye göre ilerlemeyi değerlendirmek için de metrikler kullanılabilir. Yaygın test metrikleri arasında şunlar bulunur:

- Hazırlanan test senaryo sayısının planlanan sayıya oranı
- Test ortamı hazırlığında yapılan işin yüzdesi
- Test senaryosu yürütme (örn. çalıştırılan/çalıştırılmayan test senaryosu sayısı ve başarılı/başarısız test senaryoları)
- Hata ile ilgili bilgi (örn. hata yoğunluğu, bulunan ve düzeltilen hatalar, arıza oranı ve tekrar testi sonuçları)
- Testler sonucunda gereksinimlerin, risklerin ve kodun kapsam yüzdesi
- Test uzmanlarının ürüne subjektif güveni
- Test kilometre taşlarının tarihleri
- Test maliyetleri (bir sonraki hatayı bulma avantajıyla ya da bir sonraki testi çalıştırma avantajıyla karşılaştırıldığında ortaya çıkan maliyet dahil)

5.3.2 Test Raporlama (K2)

Test raporlama, test aktiviteleri ile ilgili bilgileri özetler ve bu bilgiler aşağıdakileri içerir:

- Bir test projesi boyunca nelerin meydana geldiği (örn. çıkış kriterinin karşılandığı tarihler gibi)
- Gelecekteki eylemlere yönelik önerileri ve kararları destekleyecek analiz edilmiş bilgiler ve metrikler

(örn. kalan hataların değerlendirilmesi, devam eden testlerin ekonomik avantajı, göze çarpan riskler ve test edilen yazılıma ilişkin güven seviyesi gibi)

Test özet raporunun ana hatları "Yazılım Testi Dokümantasyonu Standardı" (IEEE Std 829-1998) kapsamında yer almaktadır.

Şu durumları değerlendirmek için test sırasında ve test seviyesinin sonunda metrikler toplanmalıdır:

- Test seviyesi için test hedeflerinin yeterliliği
- Benimsenen test yaklaşımlarının yeterliliği
- Hedeflere göre testin etkinliği

5.3.3 Test Kontrol (K2)

Test kontrol, toplanan ve raporlanan bilgilerin bir sonucu olarak ortaya çıkan yönlendirmeleri ve gerçekleştirilen düzeltici eylemleri tanımlar. Eylemler tüm test işlemlerini kapsayabilir ve diğer yazılım yaşam döngüsü işlemini veya görevini etkileyebilir.

Test kontrol eylemlerinin örnekleri şöyledir:

- Test gözetiminden alınan bilgilere dayanarak kararlar verme
- Tahmin edilen bir risk oluştuğunda (örn. yazılımın geç teslimi) testleri yeniden önceliklendirme
- Test ortamının elverişli olması veya olmaması nedeniyle test zaman çizelgesini değiştirme
- Sürüme kabul edilmeden önce yazılımcı tarafından düzeltmelerin yeniden test edilmesini gerektiren (onaylama testi) giriş kriterini belirleme

5.4 Yapılandırma Yönetimi (K2)	*10 dakika*

Terimler

Yapılandırma yönetimi, versiyon kontrolü

Genel Bilgi

Yapılandırma yönetiminin amacı, proje ve yazılım yaşam döngüsü boyunca yazılıma ait ürünlerin (bileşenler, veri ve dokümantasyon) bütünlüğünü sağlamak ve korumaktır.

Test etme konusunda yapılandırma yönetimi aşağıdakileri sağlamalıdır:

- Test süreci boyunca izlenebilirliğin korunması için tüm test yazılımı öğeleri tanımlanır, versiyonları kontrol edilir, değişiklikler izlenir, birbiriyle ve geliştirme öğeleriyle (test nesneleri) ilişkilendirilir
- Tüm dokümanlar ve yazılım öğeleri test dokümantasyonunda açık bir şekilde referans olarak verilir

Test uzmanı açısından yapılandırma yönetimi, test edilen öğeyi, test dokümanlarını, testleri ve test kuluçkalarını tanımlamaya (ve yeniden üretmeye) yardımcı olur.

Test planlama sırasında yapılandırma yönetimi prosedürleri ve altyapısı (araçlar) seçilmeli, belgelenmeli ve uyarlanmalıdır.

5.5 Risk ve Test Etme (K2)	*30 dakika*

Terimler

Ürün riski, proje riski, risk, risk bazlı test

Genel Bilgi

Risk, meydana geldiğinde istenmeyen sonuçlara ya da potansiyel bir probleme yol açabilecek bir olay, tehlike, tehdit veya durumun olasılığı olarak tanımlanabilir. Risk seviyesi, istenmeyen olayın olma ihtimali ve etkisi (bu olayın neden olacağı zarar) ile belirlenebilir.

5.5.1 Proje Riskleri (K2)

Proje riskleri, projenin hedeflerine ulaşmasını engelleyebilecek risklerdir, örneğin;

- Organizasyonel faktörler:
  - Beceri, eğitim ve personel kısıtlaması
  - Personel sorunları
  - Politik sorunlar, örneğin:
    - Test uzmanlarının ihtiyaçlarını ve test sonuçlarını karşı tarafa iletmesi ile ilgili sorunlar
    - Test ve gözden geçirmelerde bulunan bilgilerin yazılım sürecinin iyileştirilmesi için kullanılamaması
    - Test ekibinden ve sürecinden yanlış beklentiler (örn. test sırasında hata bulmanın önemini takdir etmeme)
- Teknik sorunlar:
  - Doğru gereksinimleri belirleme ile ilgili problemler
  - Gereksinimlerin kısıtları karşılayamaması
  - Test ortamının zamanında hazır olmaması
  - Geç kalınmış veri dönüştürme,
  - Geç kalınmış geçiş planlaması,
  - Geç kalınmış yazılım geliştirme,
  - Veri dönüştürme/geçiş araçlarının geç test edilmesi,
  - Tasarım ve kod kalitesinin düşük olması
  - Yapılandırma ve test verisinin kalitesinin düşük olması
- Tedarikçi sorunları:
  - Tedarikçilerin kalitesiz iş üretmesi
  - Sözleşme sorunları

Riskleri analiz ederken, yönetirken ve azaltırken, test yöneticisinin iyi oluşturulmuş proje yönetimi ilkelerini izlemesi gerekir. "Yazılım Testi Dokümantasyonu Standardı" (IEEE Std 829-1998)'nın test planlarıyla ilgili kısmında risklerin ve beklenmeyen olayların bildirilmesi gerektiği vurgulanır.

5.5.2 Ürün Riskleri (K2)

Yazılımdaki potansiyel arıza alanları (gelecekteki ters gidebilecek işler veya tehlikeler) ürünün kalitesini riske attığı için ürün riskleri olarak bilinir. Bu riskler arasında şunlar bulunur:

- Arızaya eğilimli yazılımın teslim edilmesi
- Yazılımın/donanımın bir kişiye veya şirkete zarar verebilme potansiyeli
- Yazılım özelliklerinin zayıf olması (örn. fonksiyonalite, güvenilirlik, kullanılabilirlik ve performans)
- Veri bütünlüğünün ve kalitesinin zayıf olması (örn. veri geçişi sorunları, veri dönüştürme problemleri, veri taşıma problemleri, veri standartlarının ihlali)
- Amaçlanan fonksiyonlarını gerçekleştirmeyen yazılım

Riskler, testin nereden başlatılacağına ve daha fazla testin nerede yapılacağına karar vermek için kullanılır; test etme süreci istenmeyen sonuçların ortaya çıkma riskini azaltmak veya istenmeyen bir sonucun etkisini azaltmak için hayata geçirilir.

Ürün riskleri, projenin başarısı ile ilgili bir risk çeşididir. Testin risk kontrolü amacıyla yapılıyor olması önemli hata giderme ve beklenmedik durum planlarının etkinliğinin ölçülerek geri kalan riskler hakkında geri bildirim sağlar.

Test etmede risk bazlı yaklaşım, bir projenin ilk aşamalarından başlayarak ürün riskinin seviyelerini azaltmak için proaktif fırsatlar sunar. Bu sayede risklerin en baştan tanımlanmasını ve tanımlanan bu riskler kullanılarak test planlamasının, kontrolünün, analizinin, tasarımının, hazırlığının ve yürütülmesinin yapılmasını sağlar. Risk bazlı yaklaşımda tanımlanan riskler şu amaçlarla kullanılabilir:

- Uygulanacak test tekniklerini belirleme
- Gerçekleştirilecek testin derecesini belirleme
- Önemli hataları mümkün olduğunca erken bulmak için testi önceliklendirme
- Riski azaltmak için test dışı aktivitelerin (örn. yeni tasarımcılara eğitim sağlama) uygulanıp uygulanmayacağını belirleme

Risk bazlı test, riskleri ve bu risklerin ortadan kaldırılması için gerekli olan test seviyelerini belirlemek amacıyla proje paydaşlarının kolektif bilgi ve görüşleriyle oluşur.

Yazılımda arıza olasılığının minimum seviyeye indirilmesi için risk yönetimi aktiviteleri şunlara yönelik disiplinli bir yaklaşım sunar:

- Nelerin yanlış olacağını (riskler) değerlendirme (ve düzenli şekilde yeniden değerlendirme)
- Hangi risklerin önemli olduğuna karar verme
- Bu risklerle başa çıkmak için eylemleri uygulama

Test aktiviteleri yeni risklerin tanımlanmasını sağlayabilir, hangi risklerin azaltılması gerektiği konusunda yardımcı olabilir ve riskler hakkındaki belirsizlikleri azaltabilir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

5.6 Olay Yönetimi (K3)	*40 dakika*

Terimler

Olay kaydı, olay yönetimi, olay raporu

Genel Bilgi

Testin hedeflerinden biri hataları bulmak olduğundan, gerçekleşen ve beklenen çıktılar arasındaki farklılıkların ilk başta olay olarak kaydedilmesi gerekir. Bir olay araştırıldığında olayın hata olduğu ortaya çıkabilir. Olayları ve hataları ortadan kaldırmak için uygun eylemler belirlenmelidir. Olaylar ve hatalar bulunmasından, sınıflandırılmasına, düzeltilmesine ve çözüm onayına kadar izlenmelidir. Tüm olayların yönetilmesi için şirketlerin olay yönetimi süreci ve sınıflandırma kuralları oluşturması gerekir.

Olaylar, yazılım geliştirme, gözden geçirme, test ve yazılımın kullanımı sırasında ortaya çıkabilir. Olaylar, kodda veya gereksinim dokümanlarında, yazılım geliştirme dokümanlarında, test dokümanları ve kullanıcı bilgileri ("Yardım" veya kurulum kılavuzları) gibi dokümantasyon türlerinde ortaya çıkabilir.

Olay raporlarının hedefleri aşağıdaki gibidir:

- Yazılımcılara ve diğer paydaşlara problem hakkında geri bildirim sağlayarak gerektiğinde tanımlama, izolasyon ve düzeltme sağlama
- Test liderlerine, test edilen sistemin kalitesini ve testin ilerleyişini izleme yolu sunma
- Test süreç iyileştirmesi için fikirler sunma

Olay raporunun detayları şunları içerebilir:

- Yayın tarihi, yayınlayan kuruluş ve yazar
- Beklenen ve gerçekleşen sonuçlar
- Test öğesinin (yapılandırma öğesi) ve ortamın tanımlanması
- Olayın gözlemlendiği yazılım yaşam döngüsü süreci
- Kayıtlar, veritabanı dökümleri ve ekran görüntüleri de dahil olmak üzere gerçekleşen olayın yeniden üretiminin ve çözümün sağlamak için olayın detaylandırılması
- Etkinin paydaşlar çıkarları üzerindeki etki kapsamı veya derecesi
- Etkinin yazılım üzerindeki önem derecesi
- Aciliyet / düzeltme önceliği
- Olayın durumu (örn. açık, ertelenmiş, daha önceden aynısı raporlanmış, düzeltme bekleniyor, düzeltmenin tekrar testi bekleniyor, kapalı)
- Sonuçlar, öneriler ve onaylar
- Olay nedeniyle yapılan bir düzeltmeden/değişiklikten etkilenebilen diğer alanlar gibi genel sorunlar
- Olayın izole edilmesi, onarılması ve düzeltildi olarak onaylanması için olayla ilgili proje ekip üyelerinin gerçekleştirdiği eylemlerin sırası gibi değişiklik geçmişi
- Problemi ortaya çıkaran test senaryosu

Olay raporu yapısının ana hatları "Yazılım Testi Dokümantasyonu Standardı" (IEEE Std 829-1998) kapsamında yer almaktadır.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Referanslar

5.1.1 Black, 2001, Hetzel, 1988
5.1.2 Black, 2001, Hetzel, 1988
5.2.5 Black, 2001, Craig, 2002, IEEE Std 829-1998, Kaner 2002
5.3.3 Black, 2001, Craig, 2002, Hetzel, 1988, IEEE Std 829-1998
5.4 Craig, 2002
5.5.2 Black, 2001 , IEEE Std 829-1998
5.6 Black, 2001, IEEE Std 829-1998

6. Test için Araç Desteği (K2) — 80 dakika

Test için Araç Desteğine yönelik Öğrenme Hedefleri

Hedefler, her bir modülü tamamladıktan sonra neler yapabileceğinizi tanımlar.

6.1 Test Aracı Çeşitleri (K2)

ÖH-6.1.1 Farklı test aracı çeşitlerini amaçlarına ve temel test süreci ve yazılım yaşam döngüsü aktivitelerine göre sınıflandırma (K2)

ÖH-6.1.3 Test aracı terimini ve test için araç desteğinin amacını açıklama (K2) [2]

6.2 Araçların Etkin Kullanımı: Potansiyel Avantajlar ve Riskler (K2)

ÖH-6.2.1 Test otomasyonunun ve test için araç desteğinin potansiyel avantajlarını ve risklerini özetleme (K2)

ÖH-6.2.2 Test yürütme araçları, statik analiz ve test yönetim araçları ile ilgili özel durumları hatırlama (K1)

6.3 Aracın Organizasyona Tanıtılması (K1)

ÖH-6.3.1 Bir aracın organizasyona tanıtılmasındaki temel ilkeleri belirtme (K1)

ÖH-6.3.2 Aracın değerlendirilmesi için kavram kanıtının ve araç uyarlaması için pilot çalışma aşamasının amaçlarını belirtme (K1)

ÖH-6.3.3 Satınalma sonrası iyi bir destek için gerekli olan faktörleri hatırlama (K1)

6.1 Test Aracı Çeşitleri (K2)	*45 dakika*

Terimler

Yapılandırma yönetim aracı, kapsam aracı, hata ayıklama aracı, dinamik analiz aracı, olay yönetim aracı, yük testi aracı, modelleme aracı, izleme aracı, performans testi aracı, ölçüm etkisi, gereksinim yönetim aracı, gözden geçirme aracı, güvenlik aracı, statik analiz aracı, stres test aracı, test karşılaştırıcı, test verisi hazırlama aracı, test tasarım aracı, test kuluçkası, test yürütme aracı, test yönetim aracı, birim testi çerçevesi aracı

6.1.1 Test için Araç Desteği (K2)

Test araçları testi destekleyen bir veya daha fazla aktivite için kullanılabilir. Bu araçlar arasında şunlar bulunur:

1. Test yürütme araçları, test verisi oluşturma araçları ve sonuç karşılaştırma araçları gibi doğrudan testte kullanılan araçlar
2. Testler, test sonuçlarını, verileri, gereksinimleri, olayları, hataları vb. yönetmek için kullanılanlar gibi test süreçlerini yönetmede ve test yürütmeyi raporlama ve monitörlemede yardımcı olan araçlar
3. Gözlemlerde kullanılan araçlar (örn. bir uygulama için dosya etkinliğini monitörleyen araçlar)
4. Teste yardımcı olan tüm araçlar (bu bağlamda hesap tablosu araçları da bir çeşit test aracıdır)

Test için araç desteği, projenin bağlamına göre aşağıdaki amaçlardan birine veya daha fazlasına sahip olabilir:

- Tekrar eden görevleri otomasyona geçirerek veya test planlama, test tasarımı, test raporlama ve monitörleme gibi manuel test işlemlerini destekleyerek test işlemlerinin verimliliğini artırma
- Manuel olarak yapıldığında önemli oranda kaynak gerektiren işlemleri otomasyona geçirme (örn. statik test)
- Manuel olarak yürütülemeyen işlemleri otomasyona geçirme (örn. istemci-sunucu uygulamalarının geniş ölçekli performans testleri)
- Testin güvenilirliğini artırma (örn. büyük veri karşılaştırmalarının otomasyona geçirilmesi veya davranışın simülasyonu)

"Test çerçevesi" terimi ayrıca endüstride de en az üç anlamda kullanılmaktadır:

- Test araçları oluşturmak için kullanılabilen yeniden kullanılır ve kapsamlı test kütüphaneleri (test kuluçkaları adı da verilir)
- Test otomasyonunun tasarım çeşidi (veri güdümlü, aksiyon kelimesi güdümlü test)
- Genel test yürütme süreci

Bu ders programının amacına yönelik olarak "test çerçeveleri" terimi, Kısım 6.1.6'da açıklandığı gibi ilk iki anlamında kullanılmıştır.

6.1.2 Test Aracı Sınıflandırması (K2)

Farklı test yaklaşımlarını destekleyen birçok araç bulunmaktadır. Araçlar çeşitli kriterlere göre sınıflandırılabilir: amaç, lisanslı / ücretsiz / açık kaynak kodlu / paylaşımlı, kullanılan teknoloji v.b. Bu ders programında araçlar destekledikleri test aktivitelerine göre sınıflandırılmaktadır.

Bazı araçlar yalnızca bir aktiviteyi desteklerken, diğerleri birden fazla aktiviteyi destekleyebilir, ancak en yakından ilişkili oldukları aktiviteler altında sınıflandırılır. Tek bir sağlayıcıdan gelen araçlar, özellikle de birlikte çalışmak için tasarlanmış olanlar tek bir grupta toplanabilir.

Bazı araç çeşitleri olumsuz yönde etkili olabilir ve testin gerçekleşen çıktısını etkileyebilir. Örneğin, araç tarafından yürütülen ekstra işlemler nedeniyle yazılımın gerçek çalışma davranışı ile test sırasında gözlemlenen davranış arasında zamanlama farklı olabilir veya farklı bir kod kapsamı ölçüsü ortaya çıkabilir. Gerçek davranış ile test davranışı arasında fark yaratan bu duruma ölçüm etkisi adı verilir.

Bazı araçlar ise test uzmanlarından daha çok yazılımcılara destek sağlayabilir. (örn. bileşen ve bileşen entegrasyon testi sırasında kullanılan araçlar). Bu tür araçlar aşağıdaki listede "(D)" ile işaretlenmiştir.

6.1.3 Test Yönetimi ve Testler için Araç Desteği (K1)

Yönetim araçları tüm yazılım yaşam döngüsü boyunca tüm test aktivitelerine uygulanır.

Test Yönetim Araçları

Bu araçlar testlerin yürütülmesi, hataların izlenmesi ve gereksinimlerin yönetilmesi için arayüzler sağlar ve niceliksel analizler ile test nesnelerinin raporlanmasını destekler. Ayrıca test nesnelerinin gereksinimlere karşı izlenebilirliğini destekler ve bağımsız bir versiyon kontrol özelliği veya harici bir araca arayüz içerebilir.

Gereksinim Yönetim Araçları

Bu araçlar, gereksinim komutlarını, gereksinimlerin niteliklerini (öncelik dahil) içerir, takip numaraları veri ve testlerle gereksinimlere arasında izlenebilirliği destekler. Bu araçlar ayrıca tutarsız veya eksik gereksinimleri yakalamaya da yardımcı olabilir.

Olay Yönetim Araçları (Hata Takip Araçları)

Bu araçlar; hata, arıza, değişiklik istekleri veya algılanan problemler ve anomaliler gibi olay raporlarını saklar ve yönetir; isteğe bağlı olarak istatistiksel analiz desteğiyle olayların yaşam döngüsünü yönetmeye yardımcı olur.

Yapılandırma Yönetim Araçları

Bunlar tam anlamıyla test aracı olmasalar da, özellikle işletim sistemi versiyonları, derleyiciler, tarayıcılar vb. bağlamında birden fazla donanım/yazılım ortamı yapılandırırken test yazılımının ve ilgili yazılımın depolanması ve versiyon yönetimi için gereklidir.

6.1.4 Statik Test için Araç Desteği (K1)

Statik test araçları, geliştirme sürecinde erken safhalarda daha fazla hata bulmanın uygun maliyetli bir yolunu sunar.

Gözden Geçirme Araçları

Bu araçlar, gözden geçirme süreçlerinde, kontrol listelerinde, gözden geçirme kılavuzlarında destek sağlar ve gözden geçirme yorumlarını saklamak ve iletmek, ayrıca hataları ve eforu rapor etmek için kullanılır. Büyük veya coğrafi olarak farklı alanlarda bulunan ekipler için gözden geçirme süreçlerine yardımcı olur.

Statik Analiz Araçları (D)

Bu araçlar kodlama standartlarını (güvenli kodlama dahil), iç yapı mimarisinin analizini sağlayarak dinamik testten önce yazılımcıların ve test uzmanlarının hataları bulmasını sağlar. Koda yönelik metrikler (örn. karmaşıklık) sağlayarak planlamada ve risk analizinde de yardımcı olurlar.

Modelleme Araçları (D)

Bu araçlar, tutarsızlıkları belirleyip hataları bularak yazılım modellerini (örn. ilişkisel veritabanı için fiziksel veri modeli PDM) doğrulamak için kullanılır. Bu araçlar, modele bağlı olarak test senaryoları üretmede kullanılabilir.

6.1.5 Test Gereksinimi için Araç Desteği (K1)

Test Tasarım Araçları

Bu araçlar, gereksinimlerden, grafik kullanıcı arayüzlerinden, tasarım modellerinden (durum, veri veya nesne) veya koddan test girdileri veya yürütülebilir testler ve/veya test sonucunu bilen (test oracle) oluşturmak için kullanılır.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Test Verisi Hazırlama Araçları
Test verisi hazırlama araçları, maskeleme özellikleri sayesinde veri güvenliğini sağlayarak test verisini hazırlamak için veritabanlarını, dosyaları veya veri alışverişlerini kullanır.

6.1.6 Test Yürütme ve Kayıt Tutma için Araç Desteği (K1)

Test Yürütme Araçları
Bu araçlar, test komut dosyası dili kullanımı yoluyla saklanan girdileri ve beklenen çıktıları kullanarak testlerin otomatik veya yarı otomatik olarak yürütülmesini sağlar ve genellikle her bir test koşumu için bir test kaydı tutar. Bu araçlar ayrıca testleri kaydetmek için kullanılabilir ve genellikle verilerin parametrik hale getirilmesi ve testlerdeki diğer özelleştirmeler için test komut dosyası dillerini veya GUI tabanlı yapılandırmaları destekler.

Test Kuluçkası / Birim Testi Çerçevesi Araçları (D)
Birim testi kuluçkası veya çerçevesi, sahte nesnelerin taklit uygulama veya sürücü olarak sağlanması yoluyla test nesnesinin çalışacağı ortamı simüle ederek bileşenlerin ya da sistem bölümlerinin test edilmesini kolaylaştırır.

Test Karşılaştırıcılar
Test karşılaştırıcılar; dosyalar, veritabanları veya test sonuçları arasındaki farkları belirler. Test yürütme araçları genellikle dinamik karşılaştırıcılar içerir, ancak koşturulma sonrası karşılaştırma ayrı bir karşılaştırma aracıyla yapılabilir. Bir test karşılaştırıcı, özellikle otomasyonda bir test sonucunu bilen olarak kullanabilir.

Kapsam Ölçüm Araçları (D)
Bu araçlar, koda dahil olarak veya olmayarak, bir dizi testle çalıştırılmış kodun yüzdesini ölçer (örn. komutlar, dallar veya kararlar ve bir modül ya da fonksiyon çağrıları).

Güvenlik Test Araçları
Bu araçlar yazılımın güvenlik özelliklerini değerlendirmek için kullanılır. Bu işlem, veri gizliliğini, bütünlüğünü, kimlik doğrulamasını, yetkilendirmesini, elverişliliğini ve geri çevrilmeme durumunu ele almak için yazılımın becerisini değerlendirmeyi içerir. Güvenlik araçları çoğunlukla belirli bir teknolojiye, platforma ve amaca odaklanır.

6.1.7 Performans ve Monitörleme için Araç Desteği (K1)

Dinamik Analiz Araçları (D)
Dinamik analiz araçları yalnızca yazılım yürütülürken görülebilen zamana (performans gibi) veya bellek sızıntıları gibi hataları bulur. Genellikle bileşen ve bileşen entegrasyon testinde kullanılır.

Performans Testi / Yük Testi / Stres Testi Araçları
Performans testi araçları, eş zamanlı kullanıcı sayısı, artış deseni, sıklık ve ilgili işlem yüzdesi bağlamında bir yazılımın simüle edilen kullanım koşulları altında nasıl davranacağını monitörler ve raporlar. Yük simülasyonu, çeşitli test makineleri üzerinden önceden belirlenen bir dizi işlemi gerçekleştiren sanal kullanıcılar oluşturarak elde edilir.

İzleme Araçları
İzleme araçları sürekli olarak belirli sistem kaynaklarının kullanımını analiz eder, doğrular ve raporlar, ayrıca olası hizmet problemleri ile ilgili uyarılar verir.

6.1.8 Özel Test İhtiyaçları için Araç Desteği (K1)

Veri Kalitesi Değerlendirmesi

Veri, veri dönüştürme/geçiş projeleri ve veri ambarı benzeri uygulamalar ve projelerin merkezinde bulunur. Bu tür projelerde, işlenen verilerin doğru ve eksiksiz olduğundan ve proje ihtiyaçlarına uyduğundan emin olmak amacıyla veri dönüştürme ve geçiş kurallarını gözden geçirmek ve doğrulamak üzere araçların veri kalitesi değerlendirmesi için kullanılması gerekir.

Kullanılabilirlik testleri için de kullanılan test araçları bulunmaktadır.

6.2 Araçların Etkin Kullanımı: Potansiyel Avantajlar ve Riskler (K2)	*20 dakika*

Terimler

Veri güdümlü test, aksiyon kelimesi güdümlü test, test komut dosyası

6.2.1 Teste Yönelik Araç Desteğinin Potansiyel Avantajları ve Riskleri (tüm araçlar için) (K2)

Sadece bir aracı satın almak veya kiralamak bu aracın başarılı olacağı anlamına gelmez. Gerçek ve kalıcı avantajlar elde etmek amacıyla her araç çeşidi için ek çaba sarf edilmesi gerekebilir. Testte araç kullanımının potansiyel avantajları ve fırsatları vardır, ancak riskleri de bulunmaktadır.

Araç kullanımının potansiyel avantajları şunları içerir:

- Tekrar eden işler azaltılır (regresyon testlerini koşma, aynı test verisini tekrar girme ve kod standartlarına göre kontrol etme gibi)
- Daha fazla tutarlılık ve tekrar edilebilirlik (aynı sırada, aynı sıklıkta bir araç tarafından yürütülen testler ve gereksinimlerden türetilen testler)
- Hedef değerlendirmesi (statik ölçümler, kapsam)
- Testler veya test aktivitesi ile ilgili bilgilere kolay erişim (test ilerlemesi, olay oranları ve performans hakkında istatistikler ve grafikler)

Araç kullanımının riskleri:

- Araçtan gerçek dışı beklentiler (fonksiyonalite ve kullanım kolaylığı dahil)
- Aracın ilk kez uygulanmasında harcanan zamanı, maliyeti ve çabayı göz ardı etme (eğitim ve alınacak danışmanlık dahil)
- Aracın sağlayacağı önemli ve sürekli avantajları elde etmek için gerekli süreyi ve çabayı göz ardı etme (test sürecinde değişiklik yapma gerekliliği ve aracın kullanım şeklinde sürekli iyileştirme için harcanacak zaman)
- Aracın üreteceği test varlıklarını yönetmek için gereken çabayı göz ardı etme
- Araca çok fazla güvenme (manuel testin daha iyi olacağı yerlerde otomatik test kullanmaya çalışma veya test tasarımını değiştirme)
- Aracın ürettiği test varlıklarının versiyon kontrolünü ihmal etme
- Gereksinim yönetim araçları, versiyon kontrol araçları, olay yönetimi araçları, hata takip araçları ve çeşitli tedarikçilerden alınan araçlar gibi önemli araçlar arasındaki entegrasyon ve birlikte çalışabilirlik sorunlarını görmezden gelme
- Araç tedarikçisinin sektörden çıkması, aracı kullanımdan kaldırması veya aracı farklı bir tedarikçiye satması riski
- Destek, yükseltme ve hata düzeltmeleri ile ilgili tedarikçiden yetersiz yanıt alma
- Açık kaynaklı / ücretsiz test otomasyon araçlarında aracı geliştirmekte olan ekibin projeyi askıya alma riski
- Yeni bir teknolojiyi veya platformu desteklememe gibi öngörülemeyen riskler

6.2.2 Bazı Test Otomasyon Aracı Çeşitleri ile İlgili Özel Durumlar (K1)

Test Yürütme Araçları

Test yürütme araçları, test komut dosyalarını kullanarak test nesnelerini yürütür. Bu tür araçlardan fayda sağlanması için genellikle büyük ölçüde çaba sarf edilmesi gerekir.

Manuel yapılan test adımlarının kaydedilerek testlerin kaydedilmesi kolay ve ilgi çekici görülebilir, ancak bu yaklaşım birçok test senaryosunun otomasyona geçirilmesi için uygun değildir. Oluşturulan her bir test komut dosyası ardışık ilerleyen adımlardan oluştuğu için bir adımda test sırasında yaşanan sorun test komut dosyasının ilerleyişini durdurabilir veya kararsızlaştırabilir.

Veri güdümlü test yaklaşımı test girdilerini (veriler) genellikle bir tabloda tutar ve girdileri okuyabilecek daha genel bir test komut dosyası kullanır. Böylece aynı test komut dosyası farklı verilerle yürütülmüş olur. Test uzmanları kullanılan test komut dosyası dilini bilmese bile yeni test verileri oluşturarak yeni test senaryoları oluşturabilir.

Veri güdümlü tekniklerde uygulanan başka teknikler de vardır: tabloya doğrudan gömülen veri kombinasyonları yerine, yürütme esnasında dinamik olarak yapılandırılabilir parametrelere dayanan algoritmalar kullanılarak veriler oluşturulur ve uygulamaya girdi olarak sağlanır. Örneğin bir araç rastgele bir kullanıcı kimliği oluşturan bir algoritma kullanabilir.

Aksiyon kelimesi güdümlü test yaklaşımında ise tablolarda test otomasyon aracının gerçekleştirmesi istenilen eylemleri açıklayan anahtar sözcükler (ayrıca aksiyon sözcüğü adı da verilir) ve test verileri yer alır. Test uzmanları (test komut dosyasının dilini bilmeseler bile), anahtar kelimeleri kullanarak yeni testler tanımlayabilir.

Her iki yaklaşımda da test komut dosyasının dilinde geliştirme yapmak için teknik uzmanlık gerekir.

Kullanılan yaklaşım ne olursa olsun her bir test için beklenen sonuçların daha sonra karşılaştırılmak üzere saklanması gerekir.

Statik Analiz Araçları

Kaynak koda uygulanan statik analiz araçları kodlama standartlarına uygunluğu kontrol edebilir. Var olan koda uygulandığında çok fazla miktarda uyarı mesajı üretebilir. Uyarı mesajları kodun yürütülebilir bir programa dönüştürülme işlemini engellemez, ancak kodun bakımının gelecekte daha kolay olması için bu uyarı mesajları dikkate alınmalıdır. Analiz araçlarında filtrelemeler yapılarak, bazı mesajların dahil edilmemesi etkin bir yaklaşımdır.

Test Yönetim Araçları

Test yönetim araçları, organizasyonun ihtiyaçlarına uygun bilgiler ve raporlar üretmek için diğer araçlara veya yazılımlara entegre arayüzlere sahip olması gerekir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

6.3 Aracın Kuruluşa Tanıtılması (K1)	*15 dakika*

Genel Bilgi

Bir organizasyona yönelik araç seçme ile ilgili temel adımlar şöyledir:

- Organizasyonel olgunluğun, güçlü alanların ve zayıf alanların değerlendirilmesi ve araçlar tarafından desteklenebilecek iyileştirilmiş bir test süreci için fırsatların tanımlanması
- Net gereksinimlere ve hedef kriterlerine göre değerlendirme
- Alınması planlanan test otomasyon aracının mevcut alt yapı ve test edilecek yazılımla olan uyumunun gözlemlenmesi için deneme çalışmasının yapılması, eğer bu otomasyon aracı alınırsa yapılması gereken değişikliklerin belirlenmesi
- Otomasyon aracı tedarikçisinin (eğitim, destek ve lisanslama dahil) veya ticari olmayan araçlar olması durumunda hizmet desteği sağlayıcılarının değerlendirilmesi
- Aracın kullanımında koçluk ve danışmanlık için şirket içi gereksinimlerin belirlenmesi
- Mevcut test ekibinin test otomasyonu becerilerini göz önünde bulundurarak eğitim ihtiyaçlarının değerlendirilmesi
- Bir iş senaryosuna dayanarak fayda-maliyet oranının hesaplanması

Seçilen bir test otomasyon aracının organizasyona tanıtılması pilot bir projeyle başlar ve bu proje aşağıdaki hedeflere sahiptir:

- Araç hakkında daha fazla ayrıntıyı öğrenme
- Aracın var olan süreçlere ve uygulamalara uygunluğunu değerlendirme ve değiştirilmesi gerekenleri belirleme
- Aracın ve test varlıklarının kullanılması, yönetilmesi, saklanması ve bakımının yapılması ile ilgili standart yöntemlere karar verme (örn. dosyalar ve test senaryoları için adlandırma kurallarına karar verme, kitaplıklar oluşturma ve test gruplarının modülerliğini belirleme)
- Avantajlara uygun bir maliyete ulaşılıp ulaşılmayacağını değerlendirme

Bir aracın organşzasyonda yaygınlaştırılması için başarı faktörleri:

- Aracı organizasyona aşamalı şekilde yaygınlaştırılması
- Aracın kullanımına uyacak şekilde süreçleri uyarlama ve iyileştirme
- Yeni kullanıcılar için eğitim ve koçluk/danışmanlık sağlama
- Kullanımla ilgili kılavuz bilgileri belirleme
- Canlı esnasında kullanım bilgilerini toplamak için yöntemler uyarlama
- Araç kullanımını ve avantajlarını monitörleme
- Test ekibine destek sağlama
- Tüm ekiplerden edinilen tecrübeleri toplama

Referanslar

6.2.2 Buwalda, 2001, Fewster, 1999
6.3 Fewster, 1999

7. Referanslar

Standartlar

ISTQB Glossary of Terms used in Software Testing Version 2.1

[CMMI] Chrissis, M.B., Konrad, M. and Shrum, S. (2004) CMMI, Guidelines for Process Integration *and Product Improvement*, Addison Wesley: Reading, MA Bkz. Kısım 2.1

[IEEE Std 829-1998] IEEE Std 829™ (1998) IEEE Standard for Software Test Documentation, Bkz. Kısım 2.3, 2.4, 4.1, 5.2, 5.3, 5.5, 5.6

[IEEE 1028] IEEE Std 1028™ (2008) IEEE Standard for Software Reviews and Audits, Bkz. Kısım 3.2

[IEEE 12207] IEEE 12207/ISO/IEC 12207-2008, Software life cycle processes, Bkz. Kısım 2.1

[ISO 9126] ISO/IEC 9126-1:2001, Software Engineering – Software Product Quality, Bkz. Kısım 2.3

Kitaplar

[Beizer, 1990] Beizer, B. (1990*) Software Testing Techniques* (2nd edition), Van Nostrand Reinhold: Boston
Bkz. Kısım 1.2, 1.3, 2.3, 4.2, 4.3, 4.4, 4.6

[Black, 2001] Black, R. (2001) *Managing the Testing Process* (3rd edition), John Wiley & Sons: New York
Bkz. Kısım 1.1, 1.2, 1.4, 1.5, 2.3, 2.4, 5.1, 5.2, 5.3, 5.5, 5.6

[Buwalda, 2001] Buwalda, H. et al. (2001) *Integrated Test Design and Automation*, Addison Wesley: Reading, MA
Bkz. Kısım 6.2

[Copeland, 2004] Copeland, L. (2004) *A Practitioner's Guide to Software Test Design*, Artech House: Norwood, MA
Bkz. Kısım 2.2, 2.3, 4.2, 4.3, 4.4, 4.6

[Craig, 2002] Craig, Rick D. and Jaskiel, Stefan P. (2002) *Systematic Software Testing*, Artech House: Norwood, MA
Bkz. Kısım 1.4.5, 2.1.3, 2.4, 4.1, 5.2.5, 5.3, 5.4

[Fewster, 1999] Fewster, M. and Graham, D. (1999) *Software Test Automation*, Addison Wesley: Reading, MA
Bkz. Kısım 6.2, 6.3

[Gilb, 1993]: Gilb, Tom and Graham, Dorothy (1993) *Software Inspection*, Addison Wesley: Reading, MA
Bkz. Kısım 3.2.2, 3.2.4

[Hetzel, 1988] Hetzel, W. (1988) *Complete Guide to Software Testing*, QED: Wellesley, MA Bkz. Kısım 1.3, 1.4, 1.5, 2.1, 2.2, 2.3, 2.4, 4.1, 5.1, 5.3

[Kaner, 2002] Kaner, C., Bach, J. and Pettticord, B. (2002) *Lessons Learned in Software Testing*, John Wiley & Sons: New York
Bkz. Kısım 1.1, 4.5, 5.2

[Myers 1979] Myers, Glenford J. (1979) *The Art of Software Testing*, John Wiley & Sons: New York
Bkz. Kısım 1.2, 1.3, 2.2, 4.3

[van Veenendaal, 2004] van Veenendaal, E. (ed.) (2004) *The Testing Practitioner* (Chapters 6, 8, 10), UTN Publishers: The Netherlands
Bkz. Kısım 3.2, 3.3

8. Ek A – Ders Programı Genel Bilgi *Dokümanın Geçmişi*

Bu doküman, International Software Testing Qualifications Board (ISTQB) tarafından atanan üyelerden oluşan bir çalışma grubu tarafından 2004 ile 2011 yılları arasında hazırlanmıştır. Öncelikle seçilen bir gözden geçirme paneli, ardından da uluslararası yazılım testi topluluğu arasından seçilen temsilciler tarafından gözden geçirilmiştir. Bu belgenin oluşturulmasındaki kurallar Ek C'de gösterilmektedir.

Bu doküman, ISTQB tarafından onaylanan ilk seviye uluslararası nitelik olan, Yazılım Testinde Uluslararası Temel Seviye Sertifikaya yönelik bir ders programıdır (www.istqb.org).

Temel Seviye Sertifikasyonun Hedefleri

- Yazılım mühendisliğinde test etme konusunda kabul görmek
- Test uzmanlarının kariyerlerinin gelişimi için standart bir çerçeve sağlamak
- Profesyonel nitelikteki test uzmanlarının işverenler, müşteriler ve meslektaşları tarafından kabul görmesini sağlamak ve test uzmanlarının profilini geliştirmek
- Tüm yazılım mühendisliği disiplinlerinde tutarlı ve iyi test uygulamalarını sağlamak
- Endüstriyle ilgili ve endüstri açısından değerli test konularını tanımlamak
- Yazılım tedarikçilerinin sertifikalı test uzmanları çalıştırmalarını ve böylece test uzmanı çalıştırma ilkeleri ile ilgili tanıtım yaparak rakipleri üzerinde ticari avantaj elde etmelerini sağlamak
- Test uzmanlarına ve test etme konusuyla ilgili olanlara konuyla ilgili uluslararası olarak tanınan bir nitelik elde etme fırsatı sağlamak

Uluslararası Hedefler (Kasım 2001'de Sollentuna'daki ISTQB toplantısından uyarlanmıştır)

- Farklı ülkelerdeki test etme becerilerini karşılaştırabilmek
- Test uzmanlarının ülke sınırlarını daha kolay şekilde aşmalarını sağlamak
- Çok uluslu/uluslararası projelerde test ile ilgili ortak bir anlayış geliştirmek
- Dünya çapında nitelikli test uzmanı sayısını artırmak
- Ülkeye özel yaklaşımlardan daha çok uluslararası tabanlı bir girişim olarak daha fazla etki/değer elde etmek
- Ders programı ve terminoloji yoluyla ortak bir uluslararası anlayış ve bilgi tabanı geliştirme ve tüm katılımcıların test etme ile ilgili bilgi seviyelerini artırmak
- Test etmeyi daha fazla ülkede bir meslek olarak kabul ettirmek
- Test uzmanlarının ana dillerinde kabul gören bir nitelik elde etmelerini sağlamak
- Ülkeler arasında bilgi ve kaynak paylaşımını sağlamak
- Birçok ülkeden katılım sayesinde test uzmanlarının ve bu niteliğin uluslararası alanda tanınmasını sağlamak

Bu Seviye için Giriş Gereksinimleri

ISTQB Temel Seviye Sertifika sınavına girmek için giriş kriteri, adayların yazılım testine ilgilerinin olmasıdır. Ayrıca adayların şu özelliklere sahip olması önerilir:

- Yazılım geliştirme veya yazılım testinde en az minimum seviyede tecrübesinin olması, örneğin sistem ya da kullanıcı kabul testi uzmanı veya yazılım geliştirici olarak altı aylık tecrübe

- ISTQB standartları tarafından kabul gören bir kursa katılmış olması (ISTQB tarafından tanınan Ulusal Kurullardan birinde).

Yazılım Testinde Temel Sertifikasyonun Alt Yapısı ve Tarihçesi

Yazılım test uzmanlarının bağımsız sertifikasyonu ilk kez yazılım test kurulunun kurulmasıyla 1998'de İngiltere'de British Computer Society's Information Systems Examination Board (ISEB) tarafından başlamıştı (www.bcs.org.uk/iseb). 2002 yılında Almanya'daki ASQF bir Alman test uzmanı nitelik programı başlattı (www.asqf.de). Elinizdeki bu ders programı ISEB ve ASQF ders programlarına dayanmaktadır; yeniden düzenlenmiş, güncellenmiş ve ek içerikten oluşmaktadır ve test uzmanları için en pratik şekilde yardımcı olacak konular vurgulanmıştır.

Bu ders programı sunulmadan önce alınmış ISEB, ASQF veya ISTQB tarafından tanınan bir ulusal kuruldan alınmış sertifika ISTQB sertifikasına denk sayılacaktır. Temel seviye sertifikanın süresi dolmaz ve yenilenmesi gerekmez. Verildiği tarih sertifika üzerinde gösterilmektedir.

Katılan her bir ülkede yerel konular ISTQB tarafından tanınan ulusal bir yazılım testi kurulu tarafından kontrol edilmektedir. Ulusal kurulların görevleri ISTQB tarafından belirlenir ve ülkelere göre uyarlanır. Ülke kurullarının görevleri, eğitim sağlayıcıların akreditasyonunu ve sınavları düzenlemektir.

9. Ek B – Öğrenme Hedefleri / Kavramsal Bilgi Seviyesi

Aşağıdaki öğrenme hedefleri bu ders programına uygun olacak şekilde tanımlanmıştır. Ders programındaki her bir konu, ilgili öğrenme hedefine göre sınavda yer alacaktır.

Seviye 1: Hatırlanması gereken (K1)

Aday bir terimi veya kavramı tanır, anımsar ve hatırlar.

Anahtar sözcükler: Hatırlama, alma, anımsama, tanıma, bilme

Örnek

"Arıza" tanımını şu şekilde hatırlayabilir:

- "Son kullanıcıya veya diğer paydaşlara hizmetin iletilmemesi" veya
- "Bileşenin ya da sistemin beklenen iletiminden, hizmetinden veya sonucundan sapması"

Seviye 2: Anlamanın yeterli olduğu (K2)

Aday, konu ile ilgili terimlerin sebeplerini ve açıklamalarını seçebilir, özetleyebilir, karşılaştırabilir, sınıflandırabilir, kategorize edebilir ve test kavramı ile ilgili örnekler verebilir.

Anahtar sözcükler: Özetleme, genelleştirme, özet çıkarma, sınıflandırma, karşılaştırma, haritasını çıkarma, tezat oluşturma, örneklendirme, yorumlama, çevirme, temsil etme, çıkarım yapma, sonuç çıkarma, kategorize etme, modeller oluşturma

Örnekler

Testlerin neden mümkün olduğunca erken tasarlanması gerektiğini açıklayabilir:

- Hataların ortadan kaldırılması daha ucuz olacağı
- Öncelikle en önemli hataları bulmak

Entegrasyon ve sistem testi arasındaki benzerlikleri ve farkları açıklayabilir:

- Benzerlikler: birden fazla bileşeni test etme ve fonksiyonel olmayan kavramları test edebilme
- Farklar: entegrasyon testi arayüzlere ve etkileşimlere odaklanır; sistem testi uçtan uca süreçler gibi tüm sistem ile ilgili durumlara odaklanır

Seviye 3: Uygulama gerektiren (K3)

Aday, bir kavramın veya tekniğin doğru uygulamasını seçebilir ve bunu belirli bir bağlamda uygulayabilir.

Anahtar sözcükler: Uyarlama, yürütme, kullanma, prosedürü izleme, prosedürü uygulama

Örnek

- Geçerli ve geçersiz sınıflar için sınır değerlerini tanımlayabilir
- Tüm geçişleri kapsam içine almak için belirli bir durum geçişi diyagramından test senaryolarını seçebilir

Seviye 4: Analiz edilmesi gereken (4)

Aday, bir prosedür veya teknikle ilgili bilgileri, daha iyi anlamak için bölümlerine ayırabilir ve gerçekler ve çıkarımlar arasındaki farkı belirtebilir. Genel uygulama; bir belgeyi, yazılımı veya proje durumunu analiz etmek ve bir problemi ya da görevi çözümlemek için uygun eylemler önermektir.

Anahtar sözcükler: Analiz etme, organize etme, bağlantı bulma, entegre etme, taslağını çıkarma, çözümleme, yapılandırma, dayandırma, parçalarına ayırma yöntemiyle analiz etme, farklılıkları bulma, ayırma, odaklanma, seçme

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Örnek

- Ürün risklerini analiz etme ve önleyici ve düzeltici aktiviteleri önerme
- Bir olay raporunun hangi bölümlerinin gerçek olduğunu, hangi bölümlerinin sonuçlardan çıkarıldığını açıklama

Referans

(öğrenme hedeflerinin kavramsal seviyeleri için)

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn & Bacon

10. Ek C – ISTQB Temel Ders Programı Dahilinde Uygulanan Kurallar

Temel Ders Programı

Burada listelenen kurallar, bu ders programının geliştirilmesi ve gözden geçirilmesi sırasında kullanılmıştır. (Her bir kuraldan sonra kuralın kısaltması olarak bir "ETİKET" gösterilmektedir.)

10.1.1 Genel Kurallar

SG1. Ders programı, test konusunda sıfır ila altı ay (veya daha fazla) deneyime sahip olan insanlar tarafından anlaşılabilir ve kavranabilir olmalıdır. (6 AY)
SG2. Ders programı teoriden çok pratiğe yönelik olmalıdır. (PRATİK)
SG3. Ders programı, hedeflenen okuyucular açısından net ve açık olmalıdır. (NET)
SG4. Ders programı farklı ülkelerdeki insanlar için de anlaşılabilir olmalıdır ve farklı dillere kolayca çevrilebilmelidir. (ÇEVRİLEBİLİR)
SG5. Ders programında Amerikan İngilizcesi kullanılmalıdır. (AMERİKAN İNGİLİZCESİ)

10.1.2 Mevcut İçerik

SC1. Ders programı en yeni test kavramlarını içermeli ve yazılım testinde genel olarak kabul gören en iyi uygulamaları yansıtmalıdır. Ders programı her üç ila beş yılda bir gözden geçirilmelidir. (YENİ)
SC2. Ders programı, üç ila beş yıllık bir kullanım ömrü sağlamak için güncel pazar koşulları gibi zamanla ilgili konuları en az seviyede tutmalıdır. (KULLANIM ÖMRÜ)

10.1.3 Öğrenme Hedefleri

ÖH1. Öğrenme hedefleri, anımsanması/hatırlanması gereken öğeler (kavrama seviyesi K1), adayın kavramsal olarak anlaması gereken öğeler (K2), adayın uygulayabilmesi/kullanabilmesi gereken öğeler (K3) ve adayın bir belgeyi, yazılımı ya da proje durumunu bağlamsal olarak analiz etmek için (K4) kullanabilmesi gereken öğeler olarak ayrılmalıdır. (BİLGİ SEVİYESİ)
ÖH2. İçeriğin açıklaması öğrenme hedefleri ile tutarlı olmalıdır. (ÖH İLE TUTARLI)
ÖH3. Öğrenme hedeflerinin örneklenip daha iyi anlaşılabilmesi için her bir ana başlıkla ilgili örnek sınav soruları ders programıyla birlikte verilmelidir.. (ÖH-SINAV)

10.1.4 Genel Yapı

ST1. Ders programının yapısı net olmalı, programın herhangi bir kısmı ile sınav soruları, ders programının diğer bölümleri ve diğer ilgili belgeler arasında çapraz başvuru oluşturulabilmesine olanak sağlamalıdır.. (ÇAPRAZ BAŞVURU)
ST2. Ders programının bölümleri arasındaki içerik benzerliği minimum seviyede olmalıdır. (İÇERİK BENZERLİĞİ)
ST3. Ders programının her bir bölümü aynı yapıya sahip olmalıdır. (TUTARLI YAPI)
ST4. Ders programının her sayfasında versiyon, yayın tarihi ve sayfa numarası yer almalıdır. (VERSİYON)
ST5. Ders programı, her bir kısımda harcanacak süre ile ilgili bilgiler içermelidir (her bir konunun önemini yansıtmak amacıyla). (HARCANAN SÜRE)

Referanslar

SR1. Eğitim sağlayıcıların konu ile ilgili daha fazla bilgi edinmesini sağlamak için kavramlara yönelik kaynaklar ve referanslar verilecektir. (REFERANSLAR)
SR2. Açıkça tanımlanmış ve net kaynakların olmadığı durumlarda ders programında daha fazla detay sağlanmalıdır. Örneğin, tanımlamalar sözlük kısmında yer alır, bu durumda ders programında yalnızca terimler listelenir. (REFERANS OLMAYAN AYRINTILAR)

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Bilgi Kaynakları

Bu ders programında kullanılan terimler, Yazılım Testinde kullanılan ISTQB Terimler Sözlüğünde tanımlanmaktadır. Sözlüğün bir versiyonu www.turkishtestingboard.org adresinden edinilebilir.

Yazılım testi ile ilgili önerilen kitaplar listesi de bu ders programı ile paralel şekilde sunulmuştur. Kitap listesi, Referanslar kısmının bir bölümüdür.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

11. Ek D — Eğitim Sağlayıcılara Bildiri

Ders programındaki her bir temel konuya dakika olarak süre atanmıştır. Bunun amacı, onaylı bir kursun her bir kısmına ayrılan süre ile ilgili bilgi vermek ve her bir kısmın öğretilmesi için harcanacak yaklaşık minimum süreyi vermektir. Eğitim sağlayıcılar belirtilenden daha fazla süre harcayabilir ve adaylar da okuma ve araştırma sırasında daha fazla süre harcayabilir. Bir kurs programı, ders programı ile aynı sıralamayı izlemek zorunda değildir.

Ders programı, eğitim materyalinin hazırlanması sırasında kullanılması gereken yerleşik standartlara referansları içerir. Kullanılan her bir standart, bu ders programının güncel versiyonunda belirtilen versiyon olmalıdır. Bu ders programında referans olarak kullanılmayan diğer yayınlar, şablonlar veya standartlar da kullanılabilir veya bunlara referans verilebilir, ancak sınavda yer almaz.

Tüm K3 ve K4 Öğrenme Hedefleri, eğitim materyallerine dahil edilmesi gereken uygulamalı bir alıştırma içermelidir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

12. Ek E – Sürüm Notları

Sürüm 2010

1. Öğrenme Hedefleri (ÖH) ile ilgili değişiklikler birkaç açıklama içermektedir
    a. Aşağıdaki ÖH'ler için cümle yapıları değiştirilmiştir (ÖH içeriği ve seviyesi değişmemiştir): LO-1.2.2, LO-1.3.1, LO-1.4.1, LO-1.5.1, LO-2.1.1, LO-2.1.3, LO- 2.4.2, LO-4.1.3, LO-4.2.1, LO-4.2.2, LO-4.3.1, LO-4.3.2, LO-4.3.3, LO-4.4.1, LO-4.4.2, LO-4.4.3, LO-4.6.1, LO-5.1.2, LO-5.2.2, LO-5.3.2, LO-5.3.3, LO- 5.5.2, LO-5.6.1, LO-6.1.1, LO-6.2.2, LO-6.3.2.
    b. ÖH-1.1.5 farklı bir şekilde ifade edilmiş ve K2'ye yükseltilmiştir. Hata terimlerinin karşılaştırılması ile ilgili terimler beklenebilir.
    c. ÖH-1.2.3 (K2) eklenmiştir. İçerik 2007 ders programında zaten vardır.
    d. ÖH-3.1.3 (K2) şu anda ÖH-3.1.3 ve ÖH-3.1.4 içeriğini bir arada sunmaktadır.
    e. ÖH-3.1.4, ÖH-3.1.3'te kısmen bulunduğu için 2010 ders programından çıkarılmıştır.
    f. ÖH-3.2.1, 2010 ders programı içeriğiyle tutarlı olması için farklı bir şekilde ifade edilmiştir.
    g. ÖH-3.3.2, ÖH-3.1.2 ile tutarlı olması için düzenlendi, seviyesi K1'den K2'ye değiştirilmiştir.
    h. ÖH 4.4.4 daha anlaşılır olması için düzenlenmiş ve K3'ten K4'e değiştirilmiştir. Sebep: ÖH-4.4.4 zaten K4 seviyesinde yazılmıştı.
    i. ÖH-6.1.2 (K1) 2010 ders programından çıkarıldı ve ÖH-6.1.3 (K2) ile değiştirildi. 2010 ders programında ÖH-6.1.2 yok.
2. Sözlükteki tanımına göre test yaklaşımının tutarlı kullanımı. Test stratejisi terimi hatırlanması gereken bir terim olmayacaktır.
3. Bölüm 1.4 şu anda test esası ve test senaryoları arasındaki izlenebilirlik kavramını içermektedir.
4. Bölüm 2.x şu anda test nesnelerini ve test esasını içermektedir.
5. Tekrar testi şu anda sözlükte onaylama testi yerine geçen terimdir.
6. Veri kalitesi ve test etme ders programının çeşitli yerlerine eklenmiştir: Bölüm 2.2, 5.5, 6.1.8'deki veri kalitesi ve risk.
7. Bölüm 5.2.3 Giriş Kriteri yeni bir alt bölüm olarak eklenmiştir. Sebep: Çıkış Kriteri ile Tutarlılık (-> giriş kriteri ÖH-5.2.9'a eklendi).
8. Test stratejisi ve test yaklaşımının sözlükteki tanımları ile tutarlı kullanımı.
9. Araç açıklamaları 45 dakikalık ders için çok uzun olduğundan Bölüm 6.1 kısaltıldı.
10. IEEE Std 829:2008 sürümü piyasaya çıktı. Ders programının bu versiyonu bu yeni sürümü henüz gözden geçirmedi. Kısım 5.2, Master Test Planı belgesine referans vermektedir. Master Test Planının içeriği, belge "Test Planının" farklı planlama seviyelerini kapsama kavramını içermektedir. Test seviyelerine yönelik test planları ve birden fazla test seviyesini kapsayan proje seviyesindeki test planı oluşturulabilir. Bu ders programında ve ISTQB Sözlüğünde ikincisine Master Test Planı adı verilir.
11. Etik Kuralları CTAL'den CTFL'ye taşındı.

Sürüm 2011

"Bakım sürümü" 2011 ile yapılan değişiklikler

1. Genel: Çalışma Tarafının yerine Çalışma Grubu geçti
2. ISTQB Sözlüğü 2.1 ile tutarlı olması için art koşullar "artkoşullar" olarak değiştirildi.
3. İlk örnek: ISTQB, ISTQB® olarak değiştirildi
4. Ders Programına Giriş: Kavramsal Bilgi Seviyesi açıklamaları kaldırıldı çünkü Ek B'de yer alıyordu.

5. Kısım 1.6: "Etik Kuralları" için Öğrenme Hedefi belirleme amacı olmadığından kısmın kavramsal seviyesi kaldırıldı.
6. Kısım 2.2.1, 2.2.2, 2.2.3 ve 2.2.4, 3.2.3: Listelerdeki biçimlendirme hataları düzeltildi.
7. Kısım 2.2.2 Arıza sözcüğü "...arızaların belirli bir bileşene ..." cümlesi için doğru değildi. Bu nedenle aynı cümlede "hata" sözcüğü kullanıldı.
8. Kısım 2.3: Test Çeşitleri (K2) kısmındaki test terimleri ile ilgili test hedefleri listesinin biçimlendirmesi düzeltildi.
9. Kısım 2.3.4: Hata ayıklama açıklaması ISTQB Sözlüğü Versiyon 2.1 ile tutarlı olması için güncellendi.
10. Kısım 2.4'te "kapsamlı regresyonu da içerir" cümlesinden "kapsamlı" sözcüğü çıkarıldı, çünkü "kapsamlı" bir sonraki cümlede yazıldığı gibi değişikliğe (boyut, riskler, değer vb.) bağlıdır.
11. Kısım 3.2: Cümlenin anlamının daha açık olması için "içeren" sözcüğü çıkarıldı.
12. Kısım 3.2.1: Resmi gözden geçirme işlemleri yanlış şekilde biçimlendirildiğinden, gözden geçirme sürecinde altı yerine 12 temel işlem bulunuyordu. Altı adet olacak şekilde değiştirildi ve kısmın Ders Programı 2007 ve ISTQB İleri Seviye Ders Programı 2007 ile uyumlu olması sağlandı.
13. Kısım 4: "Geliştirilmiş" sözcüğü "belirlenmiş" sözcüğü ile değiştirildi çünkü test senaryoları belirlenir, geliştirilmez.
14. Kısım 4.2: Kara kutu ve beyaz kutu testinin nasıl tecrübeye dayalı tekniklerle bağlantılı şekilde kullanılabileceğini daha açık hale getirmek için metin değiştirildi.
15. Kısım 4.3.5 "...kullanıcıları ve sistemi içeren aktörler arasında..." cümlesi "aktörler (kullanıcılar veya sistemler) arasındaki..." olarak değiştirildi.
16. Kısım 4.3.5 alternatif yol yerine alternatif senaryo kullanıldı.
17. Kısım 4.4.2: Kısım 4.4'ün metninde "dal testi" teriminin daha iyi anlaşılması için dal testinin odak noktasını açıklayan cümle değiştirildi.
18. Kısım 4.5, Kısım 5.2.6: "experienced-based" (tecrübeli kişiye dayalı) test terimi "experience-based" (tecrübeye dayalı) test ile değiştirildi.
19. Kısım 6.1: Başlık "6.1.1 Test için Araç Desteğinin Anlamını ve Amacını Anlama (K2)", "6.1.1 Test Etme için Araç Desteği (K2)" olarak değiştirildi.
20. Kısım 7 / Kitaplar: [Black,2001]'in 2nd edition'ı yerine 3rd edition'ı listelendi.
21. Ek D: Alıştırma yapılmasını gerektiren bölümler, tüm Öğrenme Hedefleri K3 ve daha üstünün alıştırma gerektirdiği ile ilgili genel gereksinimle değiştirildi. Bu ISTQB Akreditasyon Sürecinde belirtilen bir gereksinimdir (Versiyon 1.26).
22. Ek E: Versiyon 2007 ve 2010 arasındaki değişen öğrenme hedefleri şu anda doğru şekilde listelenmektedir.

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

13. Dizin

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63

Yazılım Test ve Kalite Derneği | www.turkishtestingboard.org | info@turkishtestingboard.org
Tel: +90 212 290 76 62 | Faks: +90 212 290 76 63