Sertifikalı Test Uzmanı
Temel Seviye Ders Programı

Versiyon 2018

International Software Testing Qualifications Board

Telif Hakkı Bildirimi

Revizyon Geçmişi

Versiyon	Tarih	Notlar
ISTQB 2018	27-Nisan-2018	Aday genel yayın versiyonu
ISTQB 2018	12-Şubat-2018	Aday beta versiyonu
ISTQB 2018	19-Ocak-2018	Çapraz gözden geçirme dâhili versiyon 3.0
ISTQB 2018	15-Ocak-2018	Çekirdek Ekip düzenlemelerini içeren, çapraz gözden geçirme öncesi dâhili versiyon 2.9
ISTQB 2018	9-Aralık-2017	Alfa gözden geçirmesi 2.5 sürümü - 2.0 sürümünün teknik düzenlemesidir, yeni içerik eklenmemiştir
ISTQB 2018	22-Kasım-2017	Alfa gözden geçirmesi 2.0 sürümü - Sertifikalı Test Uzmanı Temel Seviye Ders Programı Büyük Güncellemesi 2018 - Ayrıntılar için bkz. Ek C - Sürüm Notları
ISTQB 2018	12-Haziran-2017	Alfa gözden geçirmesi sürümü - Sertifikalı Test Uzmanı Temel Seviye Ders Programı Büyük Güncellemesi 2018 - bkz. Ek C - Sürüm Notları
ISTQB 2011	1-Nisan-2011	Sertifikalı Test Uzmanı Temel Seviye Ders Programı Bakım Sürümü - Sürüm Notlarına bakınız
ISTQB 2010	30-Mar-2010	Sertifikalı Test Uzmanı Temel Seviye Ders Programı Bakım Sürümü - Sürüm Notlarına bakınız
ISTQB 2007	01-May-2007	Sertifikalı Test Uzmanı Temel Seviye Ders Programı Bakım Sürümü
ISTQB 2005	01-Tem-2005	Sertifikalı Test Uzmanı Temel Seviye Ders Programı
ASQF V2.2	Temmuz-2003	ASQF Ders Programı Temel Seviye Versiyon 2.2 "Yazılım Testleri için Temel Seviye Ders Programı"
ISEB V2.0	25-Şub-1999	ISEB Yazılım Testlerinin Temelleri Ders Programı V2.0

Önsöz

ISTQB® Sertifikalı Test Uzmanı Temel Seviye Ders Programının 2011 Türkçe versiyonunun yayınlamasının üzerinden 5 yıldan fazla zaman geçti. Bu süre zarfında hem yazılım test sektöründe çok önemli gelişmeler oldu hem de sektör Türkiye'de ve dünyada yıllık ortalama %14 oranında bir ivmeyle büyüme kaydetti. Ekim 2019 itibariyle Türkiye'de ISTQB® sınavlarına giren katılımcı sayısı 4.000'e yaklaşmış, 3.000'e yakın katılımcı da sınavlarda başarılı olarak ISTQB® Uluslararası Sertifikalı Test Uzmanı sertifikalarını almaya hak kazanmıştır. Bu rakamlar dünya genelinde 920.000'den fazla sınav ve 673.000'den fazla sertifikalı test uzmanı olarak karşımıza çıkmaktadır. Sınava giren katılımcıların şu an okumakta olduğunuz Temel Seviye Ders Programını en az bir kere okuduğu düşünülürse bu doküman dünyada en çok okunan yazılım test dokümanı olma unvanını büyük ihtimalle elinde tutmaktadır.

İşte bu doküman değişen yazılım ve yazılım test dünyasındaki son gelişmeleri yansıtmak amacıyla ISTQB® tarafından revize edilerek, Haziran 2018'de bilişim sektörünün hizmetine sunulmuştur. Dokümanda özellikle Çevik yazılım geliştirme çerçevelerinin teste etkisi göze çarpmaktadır. Yazılım Test ve Kalite Derneği çeviri çalışma grubu olarak bizler de bu değerli dokümanı sektör profesyonellerine sunmak amacıyla 2019 yılı içerisinde çalışmalarımıza başlayıp çeviriyi Aralık 2019 itibariyle bitirmiş bulunmaktayız.

2011 versiyonunun çevirisinde de belirttiğimiz üzere Türkçeleştirme çalışması yapılırken daha önceden dernek tarafından çevirisi yapılmış ISTQB® Yazılım Testi Terimler Sözlüğü¨baz alınmış ve çevirinin yazılım test sektöründe kullanılan güncel 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 göz ardı edilmemelidir. Bu kısıtlardan dolayı çevirinin yaşanan gelişmeler ışığında her zaman güncel olabilmesi için yeni gelişmeleri ve önerilerinizi info@turkishtestingboard.org e-posta adresine gönderebilirsiniz.

Yazılım Test ve Kalite Derneği kar amacı gütmemekte ve faaliyetlerini gönüllülerin katkılarıyla sağlamaktadır. Derneğin karı test etkinlikleri, test çalışmaları ve üniversite burslarına aktarılmaktadır. Bu dokümanı indirip okuyarak aslında hem kendinize hem yazılım test sektörüne hem de geleceğimiz olan gençlerimize yatırım yapmış olmaktasınız.

Çevirinin Türkiye bilişim sektörüne faydalı olması dileğiyle.

Yazılım Test ve Kalite Derneği

Aralık, 2019

Teşekkür

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

- Abdurrahman Akın
- Ayten Uzun
- Berk Dülger
- Cem Uraltaş
- Cenya Abudaram
- Ceren Şahin Gebizli
- Efsan Hazal Erdem
- Emrah Yayıcı
- Erdal Okumuş
- Gizem Taşçı
- Göktürk Erdoğan
- Halil Kiracı
- Haluk Şahin
- İsmail İntikam
- Kadir Herkiloğlu
- Kadir Semih İnce
- Koray Yitmen
- Mehmet Gök
- Meltem İpek
- Murat Yazar
- Özge Oflazoğlu
- Pelin Çakmak
- Sadık Kuzu
- Saniye Öden Kapusuz
- Serra Seren Dizen
- Sezgin Çetin
- Tenzile Gül Aparı
- Ünzile Algül
- Veysel Gökçen
- Yıldıray Elmacı

Yazılım Test ve Kalite Derneği Hakkında

(Turkish Testing Board – www.turkishtestingboard.org)

Yazılım Test ve Kalite Derneği, 2006 yılından bu yana Türkiye bilişim sektöründe yazılım testi farkındalığının artması ve gelişmesi için kar amacı gütmeden gönüllü bir şekilde aşağıdaki faaliyetleri gerçekleştirmektedir:

Uluslararası Sertifikasyon Sınavları

Dernek uluslararası ISTQB® sertifika sınavlarını gerçekleştirerek sınavlarda başarılı olan katılımcılara uluslararası geçerliliği olan sertifikalar vermektedir. 2006 yılından bu yana 4.000'e yakın test uzmanı adayı derneğe başvurarak sertifika sınavlarına girmiştir. Dernek bünyesinde Türkçe ve İngilizce olarak düzenlenmekte olan sertifika sınavları:

- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Sınavı
- ISTQB® Uluslararası Sertifikalı Çevik Test Uzmanı Sınavı
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Test Analisti Sınavı
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Teknik Test Analisti Sınavı
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Test Yöneticisi Sınavı

Uluslararası TestIstanbul Konferansları – www.testistanbul.org

Dernek, 2010 yılından beri Uluslararası TestIstanbul Konferanslarını düzenlemektedir. Geçtiğimiz 10 konferansta 50 ülkeden, 34 keynote ve 5.600'den fazla katılımcı ağırlanmıştır.

Paneller

Dernek, yazılım test sektörünün gelişimi için sektör veya konu bazlı paneller organize etmektedir. Bu panellere şu ana kadar 1.000'den fazla profesyonel katılım göstermiştir. Şimdiye kadar düzenlenen paneller:

- TestFintech,
- TestDefence,
- TestGames,
- TestInsurance,
- TestAnkara,
- TestİzmirTest Finance,

Turkey Software Quality Report (TSQR)

Dernek tarafından 2011 yılından itibaren yüzlerce bilişim profesyoneli ve akademisyeninin katılımıyla her yıl düzenlenen anket sonuçlarının değerlendirilmesiyle hazırlanan, Türkiye bilişim sektörüne yön verir nitelikte çıkarımların olduğu rapordur. İngilizce yayınlanan rapor tüm ISTQB® üye dernekleri aracılığıyla 100'den fazla ülkedeki bilişim profesyoneline ulaşmaktadır.

ISTQB® Worldwide Software Testing Practices Report (WSTPR)

ISTQB® tarafından 100'den fazla ülkeden binlerce bilişim profesyoneli ve akademisyeninin katılımıyla düzenlenen anket sonuçlarının değerlendirilmesiyle hazırlanan, dünya bilişim sektörüne yön verir nitelikte çıkarımların olduğu rapordur

Türkçeleştirme Çalışmaları

Uluslararası yazılım test terminolojisinin ülkemize kazandırılması için dernek bünyesinde yer alan gönüllü çeviri grubu ISTQB® dokümanlarının çevirisi üzerinde çalışmaktadır. Şu ana kadar çevrilen dokümanlar:

- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Ders Programı 2018
- ISTQB® Uluslararası Sertifikalı Temel Seviye Yazılım Test Uzmanı Ders Programı 2011
- ISTQB® Yazılım Testi Terimler Sözlüğü
- ISTQB® Uluslararası Sertifikalı İleri Seviye – Test Analisti Ders Programı 2012

Burslar

Dernek her yıl karından belli bir miktarı T.C. Üniversitelerinin Bilgisayar/Yazılım Mühendisliği ve Bilgisayar Programcılığı bölümlerinde okumakta olan başarılı ve ihtiyaç sahibi öğrencilere burs olarak aktarmaktadır.

İçindekiler Tablosu

Teşekkür

Bu doküman, ISTQB Genel Kurulu tarafından resmi olarak yayınlanmıştır (4 Haziran 2018).
Doküman, International Software Testing Qualifications Board'dan bir ekip tarafından hazırlanmıştır: Klaus Olsen (Başkan), Tauhida Parveen (Başkan Yardımcısı), Rex Black (Proje Yöneticisi), Debra Friedenberg, Judy McKay, Meile Posthuma, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh ve Eshraka Zakaria.

Ekip, teknik düzenlemeleri için Rex Black ve Dorothy Graham'a, önerileri ve girdileri için gözden geçirme ekibi, çapraz gözden geçirme ekibi ve Üye Kurullara teşekkür eder.

Bu ders programının gözden geçirilmesi, yorumlanması ve oylanmasında aşağıdaki kişiler yer almıştır: Tom Adams, Tobias Ahlgren, Xu Aiguo, Chris Van Bael, Katalin Balla, Graham Bath, Gualtiero Bazzana, Arne Becher, Veronica Belcher, Lars Hilmar Bjørstrup, Ralf Bongard, Armin Born, Robert Bornelind, Mette Bruhn-Pedersen, Geza Bujdoso, Earl Burba, Filipe Carlos, Young Jae Choi, Greg Collina, Alessandro Collino, Cui Zhe, Taz Daughtrey, Matthias Daigl, Wim Decoutere, Frans Dijkman, Klaudia Dussa-Zieger, Yonit Elbaz, Ofer Feldman, Mark Fewster, Florian Fieber, David Frei, Debra Friedenberg, Conrad Fujimoto, Pooja Gautam, Thorsten Geiselhart, Chen Geng, Christian Alexander Graf, Dorothy Graham, Michel Grandjean, Richard Green, Attila Gyuri, Jon Hagar, Kobi Halperin, Matthias Hamburg, Zsolt Hargitai, Satoshi Hasegawa, Berit Hatten, Wang Hongwei, Tamás Horváth, Leanne Howard, Chinthaka Indikadahena, J. Jayapradeep, Kari Kakkonen, Gábor Kapros, Beata Karpinska, Karl Kemminger, Kwanho Kim, Seonjoon Kim, Cecilia Kjellman, Johan Klintin, Corne Kruger, Gerard Kruijff, Peter Kunit, Hyeyong Kwon, Bruno Legeard, Thomas Letzkus, Alon Linetzki, Balder Lingegård, Tilo Linz, Hongbiao Liu, Claire Lohr, Ine Lutterman, Marek Majernik, Rik Marselis, Romanos Matthaios, Judy McKay, Fergus McLachlan, Dénes Medzihradszky, Stefan Merkel, Armin Metzger, Don Mills, Gary Mogyorodi, Ninna Morin, Ingvar Nordström, Adam Novak, Avi Ofer, Magnus C Ohlsson, Joel Oliviera, Monika Stocklein Olsen, Kenji Onishi, Francisca Cano Ortiz, Gitte Ottosen, Tuula Pääkkönen, Ana Paiva, Tal Pe'er, Helmut Pichler, Michaël Pilaeten, Horst Pohlmann, Andrew Pollner, Meile Posthuma, Vitalijs Puiso, Salvatore Reale, Stuart Reid, Ralf Reissing, Shark Ren, Miroslav Renda, Randy Rice, Adam Roman, Jan Sabak, Hans Schaefer, Ina Schieferdecker, Franz Schiller, Jianxiong Shen, Klaus Skafte, Mike Smith, Cristina Sobrero, Marco Sogliani, Murian Song, Emilio Soresi, Helder Sousa, Michael Sowers, Michael Stahl, Lucjan Stapp, Li Suyuan, Toby Thompson, Steve Toms, Sagi Traybel, Sabine Uhde, Stephanie Ulrich, Philippos Vakalakis, Erik van Veenendaal, Marianne Vesterdal, Ernst von Düring, Salinda Wickramasinghe, Marie Walsh, Søren Wassard, Hans Weiberg, Paul Weymouth, Hyungjin Yoon, John Young, Surong Yuan, Ester Zabar ve Karolina Zmitrowicz.

International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Basım 2018): Klaus Olsen (Başkan), Tauhida Parveen (Başkan Yardımcısı), Rex Black (Proje Yöneticisi), Dani Almog, Debra Friedenberg, Rashed Karim, Johan Klintin, Vipul Kocher, Corne Kruger, Sunny Kwon, Judy McKay, Thomas Müller, Igal Levi, Ebbe Munk, Kenji Onishi, Meile Posthuma, Eric Riou du Cosquer, Hans Schaefer, Radoslaw Smilgin, Mike Smith, Steve Toms, Stephanie Ulrich, Marie Walsh, Eshraka Zakaria ve Stevan Zivanovic. Çekirdek ekip, önerileri için gözden geçirme ekibine ve tüm Üye Kurullara teşekkür eder.

International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Basım 2011): Thomas Müller (Başkan), Debra Friedenberg. Çekirdek ekip, 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 Üye Kurullara teşekkür eder.
International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Basım 2010): Thomas Müller (Başkan), Rahul Verma, Martin Klonk ve Armin Beer. Çekirdek ekip, öneriler 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 Üye Kurullara teşekkür eder.
International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Basım 2007): Thomas Müller (Başkan), Dorothy Graham, Debra Friedenberg ve Erik van Veenendaal. Çekirdek ekip, önerileri için gözden geçirme ekibine (Hans Schaefer, Stephanie Ulrich, Meile Posthuma, Anders Pettersson ve Wonil Kwon) ve tüm Üye Kurullara teşekkür eder.

International Software Testing Qualifications Board Temel Seviye Çalışma Grubu (Basım 2005): Thomas Müller (Başkan), Rex Black, Sigrid Eldh, Dorothy Graham, Klaus Olsen, Maaret Pyhäjärvi, Geoff Thompson ve Erik van Veenendaal. Çekirdek ekip, önerileri için gözden geçirme ekibine ve tüm Üye Kurullara teşekkür eder.

0 Giriş

0.1 Bu Ders Programının Amacı

Bu ders programının amacı temel seviye uluslararası yazılım test uzmanı niteliklerine yönelik bir çerçeve oluşturmaktır. ISTQB aşağıda belirtilen kişi ve kurumlarla bu ders programını paylaşır:

1. Üye Kurullar: kendi dillerine çevirmeleri ve eğitim kurumlarını akredite etmeleri için. Üye Kurullar ders programını kendi özel dil ihtiyaçlarına göre uyarlayabilir ve yerel yayınlarına uyacak şekilde referanslar ekleyebilir.
2. Sertifikasyon kurumları: bu ders programına yönelik kendi dillerinde sınav soruları hazırlamaları için.
3. Eğitim kurumları: eğitim materyali üretmeleri ve uygun öğretim yöntemlerini belirlemeleri için.
4. Sertifika programı adayları: (bir eğitim kursunun parçası olarak veya bağımsız olarak) sertifika sınavına hazırlanmaları için.
5. Uluslararası yazılım ve sistem mühendisliği topluluğu: yazılım ve sistem test uzmanlığı mesleğini ilerletmek ve kitap ve makalelere bir temel oluşturmak için.

ISTQB, önceden ISTQB'den yazılı izin istemeleri ve almaları şartıyla, diğer kurumların bu ders programını başka amaçlarla kullanmalarına izin verebilir.

0.2 Yazılım Testinde Sertifikalı Test Uzmanı - Temel Seviye

Temel seviye ders programı yazılım testiyle uğraşan herkese yöneliktir. Buna, test uzmanları, test analistleri, test mühendisleri, test danışmanları, test yöneticileri, kullanıcı kabul testi uzmanları ve yazılımcılar gibi çeşitli rollerdeki kişiler dâhildir. Bu ders programı ayrıca ürün sahipleri, proje yöneticileri, kalite yöneticileri, yazılım geliştirme yöneticileri, iş analistleri, BT yöneticileri ve yönetim danışmanları gibi temel bir yazılım testi anlayışına sahip olmak isteyen herkes için de uygundur. Temel Seviye Sertifikasına sahip profesyoneller, ISTQB ileri seviye yazılım test ders programlarına devam edebilirler.

ISTQB Temel Seviye Genel Bakış 2018 dokümanı, aşağıdaki bilgileri içeren ayrı bir dokümandır:

- Ders programının iş hayatına yönelik çıktıları
- İş hayatına yönelik çıktılar ve öğrenme hedefleri arasındaki izlenebilirliği gösteren matris
- Ders programının özeti

0.3 Sınav Kapsamındaki Öğrenme Hedefleri ve Bilginin Bilişsel Seviyeleri

Öğrenme hedefleri iş hayatına yönelik çıktıları destekler ve Sertifikalı Test Uzmanı Temel Seviye sınavlarını hazırlamakta kullanılır.

Genel olarak, bu ders programının tüm içeriği, Giriş ve Ekler hariç olmak üzere K1 seviyesinde sınav kapsamındadır. Adaydan altı bölümden herhangi birinde belirtilen bir anahtar kelimeyi veya kavramı bilmesi veya hatırlaması istenebilir. Özel öğrenme hedeflerine yönelik bilgi seviyeleri, her bölümün başında gösterilmektedir ve aşağıdaki gibi sınıflandırılmıştır:

- K1: hatırla
- K2: anla
- K3: uygula

Öğrenme hedefleri için daha fazla ayrıntı ve örnekler Ek B'de verilmiştir.
Bölüm başlıklarının hemen altında anahtar kelimeler olarak listelenen tüm terimlerin tanımlarının öğrenme hedeflerinde açıkça belirtilmiş olmasa bile hatırlanması gerekmektedir (K1).

0.4 Temel Seviye Sertifika Sınavı

Temel Seviye Sertifika sınavı bu ders programını baz almaktadır. Sınav sorularını cevaplamak için, bu ders programının birden fazla bölümünü baz alan ders materyali gerekebilir. Giriş ve ekler hariç olmak üzere bu ders programının tüm bölümleri sınav kapsamındadır. Bu ders programına bazı standartlar, kitaplar ve diğer ISTQB ders programları referans olarak dâhil edilmiştir; ancak bunların içerikleri, bu ders programında bahsedildiği kadar sınava dahildir. Bahsedilmeyen kısım ve içerikler sınav kapsamına dâhil değildir.

Sınav çoktan seçmelidir ve sınavda 40 soru bulunmaktadır. Sınavı geçmek için soruların en az %65'inin (26 sorunun) doğru cevaplanması gerekir. Yanlış cevaplar doğruları götürmemektedir.

0.5 Akreditasyon

Bir ISTQB Üye Kurulu, ders materyali bu ders programına uygun olan eğitim kurumlarını akredite edebilir. Eğitim kurumları, akreditasyonu gerçekleştiren üye kuruldan veya kuruluştan akreditasyon rehberini edinmelidir. Akredite edilmiş bir kursun bu ders programına uygun olduğu kabul edilir ve kursun bir parçası olarak ISTQB sınavına girilmesine izin verilir.

0.6 Ayrıntı Düzeyi

Bu ders programındaki ayrıntı düzeyi, uluslararası kurs ve sınavlara olanak sağlar. Bu amaca ulaşmak için, ders programında aşağıdakilere yer verilmiştir:

- Temel seviyenin amacını tanımlayan genel öğretim hedefleri
- Öğrencilerin hatırlamaları gereken bir terimler listesi
- Her bilgi alanı için, ulaşılacak bilişsel öğrenme çıktılarını tanımlayan öğrenme hedefleri
- Genel kabul görmüş literatür veya standartlar gibi kaynaklara verilen referanslar dâhil olmak üzere temel kavramların açıklaması

Ders programı içeriği, yazılım testlerindeki tüm bilgi birikiminin bir açıklaması değildir; sadece temel seviye eğitim kurslarında ele alınacak detay seviyesini gösterir. Çevik projeler de dâhil olmak üzere tüm yazılım projelerine uygulanabilecek test kavramları ve tekniklerine odaklanır. Bu ders programı, herhangi bir yazılım geliştirme yaşam döngüsü veya yöntemi ile ilgili özel bir öğrenme hedefi içermez, ancak bu kavramların çevik projelerde, diğer döngüsel ve artımlı yaşam döngülerinde ve sıralı yaşam döngülerinde nasıl uygulandığının üzerinde durur.

0.7 Bu Ders Programı Nasıl Düzenlendi

Ders programı sınava tabi olan altı bölümden oluşmaktadır. Her bölümde yer alan en üstteki başlık o bölüm için ayrılan zamanı belirtir; zaman planlaması o bölümün geneli için verilmiştir, daha detayda bir planlama verilmemiştir. Akredite eğitim kursları için ders programı aşağıdaki gibi altı bölüme ayrılmış olup toplamda en az 16,75 saat kurs alınmasını gerektirir:

- Bölüm 1: 175 dakika Yazılım Testinin Temelleri
- Bölüm 2: 100 dakika Yazılım Geliştirme Yaşam Döngüsü Boyunca Testler
- Bölüm 3: 135 dakika Statik Testler
- Bölüm 4: 330 dakika Test Teknikleri
- Bölüm 5: 225 dakika Test Yönetimi
- Bölüm 6: 40 dakika Yazılım Testleri için Araç Desteği

1. Yazılım Testinin Temelleri	175 dakika

Yazılım Testinin Temelleri için Öğrenme Hedefleri:

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

Yazılım Testinin Temelleri için Öğrenme Hedefleri:

1.1 Yazılım Testi nedir?

FL-1.1.1 (K1) Yazılım testinin genel hedeflerini tanımlayın

FL-1.1.2 (K2) Test ile hata ayıklamanın farklarını belirtin testing - debugging

1.2 Yazılım Testi Neden Gereklidir?

FL-1.2.1 (K2) Örnekler vererek yazılım testinin neden gerekli olduğunu açıklayın

FL-1.2.2 (K2) Yazılım testi ve kalite güvence arasındaki ilişkiyi tanımlayın ve testlerin kaliteyi nasıl artırdığına dair örnekler verin

FL-1.2.3 (K2) İnsan hatası, hata ve arıza arasındaki farkı ayırt edin error, defect, failure

FL-1.2.4 (K2) Bir hatanın kök nedeni ile etkileri arasındaki farkı ayırt edin root cause

1.3 Yedi Test Prensibi

FL-1.3.1 (K2) Yedi test prensibini açıklayın

1.4 Test Süreci Test Process

FL-1.4.1 (K2) Proje bağlamının test süreci üzerindeki etkisini açıklayın

FL-1.4.2 (K2) Test sürecindeki test faaliyetlerini ve ilgili görevleri tanımlayın

FL-1.4.3 (K2) Test sürecini destekleyen çalışma ürünlerini tanımlayın

FL-1.4.4 (K2) Test esası ve test çalışma ürünleri arasında izlenebilirliğin sağlanmasının önemini açıklayın

1.5 Test Etme Psikolojisi

FL-1.5.1 (K1) Yazılım testinin başarısını etkileyen psikolojik faktörleri tanımlayın

FL-1.5.2 (K2) Test faaliyetleri için gereken bakış açısı ile yazılım geliştirme faaliyetleri için gereken bakış açısı arasındaki farkı açıklayın

1.1 Yazılım Testi Nedir?

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

Yazılım testi hakkındaki yaygın yanlış kanılardan biri yazılım testinin yazılımı çalıştırmaktan ve sonuçları kontrol etmekten ibaret olduğudur. Bölüm 1.4'te de açıklandığı gibi, yazılım testi birçok farklı faaliyeti içeren bir süreçtir; test koşumu aktivitesi (sonuçların kontrolü de dâhil olmak üzere) bu faaliyetlerden sadece bir tanesidir. Test süreci aynı zamanda test planlama, test analizi, test tasarımı, testin uyarlanması, test ilerlemesini ve sonuçlarını raporlama ve bir test nesnesinin kalitesini değerlendirme gibi faaliyetleri içerir.

Bazı testler, test edilen birimin veya sistemin çalıştırılmasını içerir; bu testlere dinamik testler denir. Diğer testler, test edilen birimin veya sistemin çalıştırılmasını gerektirmez; bu testlere ise statik testler denir. Dolayısıyla testler; gereksinimler, kullanıcı hikâyeleri ve kaynak kod gibi çalışma ürünlerinin gözden geçirilmesini de içerir.

Testler hakkındaki bir başka yanlış kanı ise testlerin tamamen gereksinimlerin doğrulanmasına, kullanıcı hikâyelerine veya diğer spesifikasyonlara odaklandıklarıdır. Her ne kadar yazılım testleri sistemin belirli gereksinimleri karşılayıp karşılamadığını kontrol etmeyi içerse de, aynı zamanda sistemin operasyonel ortamında/ortamlarında kullanıcı ve diğer paydaş ihtiyaçlarını karşılayıp karşılamadığını kontrol etmek anlamına gelen sağlama yapmayı da içerir.

Test faaliyetleri farklı yaşam döngülerinde farklı şekilde düzenlenir ve yürütülür (bkz. Bölüm 2.1).

1.1.1 Yazılım Testinin Genel Hedefleri

Herhangi bir projede testin amaçları aşağıdakileri içerebilir:

- Gereksinimler, kullanıcı hikâyeleri, tasarım ve kod gibi çalışma ürünlerini değerlendirmek ve kusurları önlemek.
- Belirtilen tüm gereksinimlerin yerine getirilip getirilmediğini doğrulamak
- Test nesnesinin eksiksiz olup olmadığının ve kullanıcıların ve diğer paydaşların beklediği şekilde çalıştığının sağlamasını yapmak
- Test nesnesinin kalite seviyesi hakkında güven oluşturmak
- Hataları önlemek
- Arızaları ve hataları tespit etmek ve yetersiz yazılım kalitesi riskini azaltmak.
- Özellikle test nesnesinin kalite seviyesiyle ilgili olarak sağlıklı kararlar almalarını sağlamak için paydaşlara yeterli bilgiyi sunmak
- Yazılımın kalitesiz olma riskini düşürmek (örneğin, canlıda meydana gelebilecek arızaların önceden tespiti)
- Sözleşmeden kaynaklanan, yasal veya düzenleyici gereksinimlere veya standartlara uymak ve/veya test nesnesinin bu gereksinimlere veya standartlara uyumluluğunu doğrulamak

Testin hedefleri, proje bağlamına, test seviyesine ve yazılım geliştirme yaşam döngüsü modeline bağlı olarak değişebilir. Örneğin bu farklılıklar arasında aşağıdakiler yer alabilir:

- Birim testi sırasındaki hedeflerden biri, olabildiğince çok sayıda arıza tespit etmek ve böylece bunlara neden olan hataların erkenden tespitini ve düzeltilmesini sağlamak olabilir. Diğer bir hedef, birim testinin kod kapsamını artırmak olabilir.
- Kabul testi sırasındaki hedeflerden biri sistemin beklendiği gibi çalıştığını ve gereksinimleri karşıladığını doğrulamak olabilir. Bu testin bir diğer hedefi, belirli bir zaman aralığında sistemin piyasaya sürülmesindeki riskler hakkında paydaşlara bilgi vermek olabilir.

1.1.2 Yazılım Testi ve Hata Ayıklama Testing and Debugging

Test etme ve hata ayıklama farklı aktivitelerdir. Testlerin yürütülmesi, yazılımdaki hataların neden olduğu arızaları gösterebilir. Hata ayıklama ise arızaların arkasında yatan hataları bulan, analiz eden ve düzelten yazılım geliştirme faaliyetidir. Daha sonra gerçekleştirilen onaylama testleri düzeltmelerin hataları giderip gidermediğini kontrol eder. Bazı durumlarda, test uzmanları başlangıçtaki testlerden ve son onaylama testinden sorumluyken, yazılımcılar hata ayıklama ve ilgili birim testlerini gerçekleştirir. Bununla birlikte, çevik yazılım geliştirmede ve diğer bazı yaşam döngülerinde test uzmanları hata ayıklama ve birim testlerine dâhil olabilirler.

ISO standardı (ISO/IEC/IEEE 29119-1), yazılım testi kavramları hakkında daha fazla bilgi vermektedir.

1.2 Yazılım Testi Neden Gereklidir?

Birimlerin, sistemlerin ve bunlarla ilgili dokümantasyonun titizlikle test edilmesi, uygulama sırasında arıza oluşması riskini düşürebilir. Hataların tespit edilmesi ve ardından düzeltilmesi, test edilen bu birimlerin veya sistemlerin kalitesine katkıda bulunur. Ek olarak, sözleşmeye bağlı veya yasal gereksinimleri veya sektöre özel standartları karşılamak için yazılım testleri gerekebilir.

1.2.1 Yazılım Testinin Başarıya Katkısı

Bilişim tarihi boyunca, yazılım ve sistemlerin uygulamaya alınması ve daha sonra var olan hatalar nedeniyle arızalara neden olması veya paydaşların ihtiyaçlarını karşılamaması oldukça sık görülmüştür. Bununla birlikte uygun test tekniklerinin kullanılması, bu tekniklerin uygun test seviyelerinde ve yazılım geliştirme yaşam döngüsünün uygun noktalarında uygulanması bu gibi sorunlu ürün teslimlerinin sıklığını azaltabilmektedir. Bazı örnekleri şunlardır:

- Gereksinimlerin gözden geçirilmesi veya kullanıcı hikâyesinin iyileştirilmesi süreçlerine test uzmanlarının katılması, bu çalışma ürünlerindeki hataların tespit edilmesini sağlayabilir. Gereksinimlerdeki hataların belirlenmesi ve giderilmesi, yanlış veya test edilemeyen fonksiyonalitenin geliştirilme riskini azaltır.
- Sistem tasarlanırken test uzmanlarının sistem tasarımcılarıyla birlikte çalışması, tarafların tasarım ve tasarımın nasıl test edileceği konusundaki anlayışını artırabilir. Bu anlayış artışı, tasarım hatalarını azaltabilir ve yapılacak testlerin erken bir aşamada belirlenmesini sağlayabilir.
- Kod geliştirme aşamasında test uzmanlarının yazılımcılarla birlikte çalışması, tarafların kod ve kodların nasıl test edileceği konusundaki anlayışını artırabilir. Bu anlayış artışı, kod ve testlerde hata çıkma riskini azaltabilir.
- Canlıya alınmadan önce test uzmanlarının yazılımı doğrulaması ve sağlamasını yapması, kaçırılması mümkün arızaları tespit edebilir ve arızalara neden olan hataları giderme sürecini destekleyebilir (hata ayıklama). Bu, yazılımın paydaş ihtiyaçlarını ve gereksinimleri karşılama olasılığını artırır.

Bu örneklere ek olarak, belirlenen test hedeflerinin gerçekleştirilmesi (bkz. Bölüm 1.1.1) genel yazılım geliştirme ve bakım başarısına katkıda bulunur.

1.2.2 Kalite Güvence ve Yazılım Testi Quality Assurance and Testing

İnsanlar testlerden bahsederken sıklıkla "kalite güvence" (veya sadece *KG*) tabirini kullansa da, kalite güvence ve test aynı şey değildir, ancak birbiriyle ilişkilidir. Daha büyük bir kavram olan kalite yönetimi bu ikisini birbirine bağlar. Kalite yönetimi, bir kurumu kalite açısından yönlendiren ve kontrol eden tüm faaliyetleri içerir.
Kalite yönetimi diğer faaliyetlerle birlikte kalite güvenceyi ve kalite kontrolünü de içerir. Kalite güvence genellikle, uygun kalite seviyelerinin elde edileceğine dair güven sağlamak için doğru süreçlere bağlı kalmaya odaklanır. Doğru süreçler doğru şekilde gerçekleştirildiğinde, bu süreçlerde oluşturulan çalışma ürünleri genellikle yüksek kalitededir ve bu da hataların önlenmesine katkıda bulunur. Ek olarak, süreçleri iyileştirmek için geriye dönük değerlendirme (GDD) toplantılarının bulgularının doğru bir şekilde uygulanması ile birlikte, hataların nedenlerini belirlemek ve gidermek için kök neden analizinin kullanılması, etkin kalite güvence için önemlidir.

Kalite kontrol, uygun kalite seviyelerinin elde edilmesini destekleyen test faaliyetleri de dâhil olmak üzere çeşitli faaliyetleri içerir. Test faaliyetleri, genel yazılım geliştirme veya bakım sürecinin bir parçasıdır. Kalite güvence, tüm sürecin doğru şekilde yürütülmesini hedeflediğinden testlerin doğru şekilde yapılmasını desteklemektedir. Bölüm 1.1.1 ve 1.2.1'de açıklandığı gibi, testler kalitenin elde edilmesine çeşitli şekillerde katkıda bulunur.

1.2.3 İnsan Hataları, Hatalar ve Arızalar

Bir kişinin yaptığı bir yazılım hatası (yanlışlık), yazılım kodunda veya ilgili başka bir çalışma ürününde bir hatanın (kusur) ortaya çıkmasına neden olabilir. Çalışma ürününde bir hata oluşmasına neden olan bir yazılım hatası, bir hatayı tetikleyebilir, bu da ilgili bir çalışma ürününde başka bir hatanın ortaya çıkmasına neden olabilir. Örneğin, gereksinim belirleme sırasında yapılan bir insan hatası, bir gereksinim hatasına neden olabilir ve bu da kodda bir hataya neden olan bir programlama hatasıyla sonuçlanabilir.

Hatalı kod çalıştırıldığında, bir arızaya neden olabilir, ancak bu durum her koşulda gerçekleşmeyebilir. Örneğin, bazı hataların, bir arızayı tetiklemesi için çok özel girdiler veya ön koşullar gerekebilir; bunun gerçekleşmesi de çok nadiren olabilir veya asla gerçekleşmeyebilir.

İnsanların yaptığı hataların birçok nedeni olabilir, örneğin:

- Zaman baskısı
- İnsani yanılma payı
- Deneyimsizlik veya yetersiz yetkinlik
- Gereksinimler ve tasarım ile ilgili yanlış iletişim de dâhil olmak üzere proje ekip üyeleri arasındaki yanlış iletişim
- Kodun, tasarımın, mimarinin, çözülecek temel problemin ve/veya kullanılan teknolojilerin karmaşıklığı
- Özellikle sistem içi ve sistemler arası etkileşimlerin sayısının fazla olduğu durumlarda, sistem içi ve sistemler arası arayüzler hakkındaki yanlış anlaşılmalar
- Yeni, henüz tecrübe edilmemiş teknolojiler

Koddaki hatalardan kaynaklanan arızalara ek olarak, çevresel koşullardan kaynaklanan arızalar da olabilir. Örneğin, radyasyon, elektromanyetik alanlar ve kirlilik, üretici yazılımında hatalara neden olabilir veya donanım koşullarını değiştirerek yazılımın çalışmasını etkileyebilir.
Beklenmedik test sonuçlarının tamamı arıza değildir. Yanlış pozitifler, testlerin uygulanma şeklindeki hatalardan veya test verilerindeki, test ortamındaki veya diğer test yazılımındaki hatalar nedeniyle veya diğer nedenlerden dolayı ortaya çıkabilir. Benzer hataların veya insan hatalarının yanlış negatiflere neden olduğu bunun tersi durumlar da ortaya çıkabilir. Yanlış negatifler, bulunması gereken hataları bulamayan testlerin sonuçlarıdır; yanlış pozitifler hata olarak rapor edilmesine rağmen, gerçekte hata değillerdir.

1.2.4 Defects, Root Causes and Effects — Hatalar, Kök Nedenleri ve Etkileri

Hataların kök nedenleri, hataların oluşmasına sebep olan en baştaki eylemler veya koşullardır. Hatalar, kök nedenlerini belirlemek için analiz edilebilir, böylece gelecekte benzer hataların ortaya çıkma olasılığının azaltılması hedeflenir. En belirgin kök nedenlere odaklanan kök neden analizi, gelecekte önemli sayıda hatanın ortaya çıkmasını önleyen süreç iyileştirmelerini sağlayabilir.

Örneğin, hatalı bir kod satırı nedeniyle hatalı faiz ödemelerinin müşteri şikâyetleriyle sonuçlandığını varsayalım. Hatalı kod, ürün sahibinin faizin nasıl hesaplanacağını yanlış anlaması nedeniyle muğlâk yazılan bir kullanıcı hikâyesinden kaynaklanmıştır. Faiz hesaplamalarında büyük oranda hata bulunuyorsa ve bu hataların kök nedenleri de benzer yanlış anlamalardan kaynaklanıyorsa, gelecekte oluşabilecek buna benzer hataları önlemek için ürün sahipleri faiz hesaplamaları konusunda eğitilebilir.

Bu örnekte, faizin yanlış ödenmesi arıza, müşteri şikâyetleri ise bu arızanın yarattığı etkilerdir. Koddaki yanlış hesaplama bir hatadır ve asıl hatadan, kullanıcı hikâyesindeki muğlaklıktan kaynaklanmıştır. Asıl hatanın temel nedeni, ürün sahibinin bilgi eksikliğidir; bu eksiklik, ürün sahibinin kullanıcı hikayesini yazarken hata yapmasıyla sonuçlanmıştır. Kök neden analizi süreci, ISTQB-ETM Uzman Seviye Test Yönetimi Ders Programı ve ISTQB-EITP Uzman Seviye Test Sürecini İyileştirme Ders Programında detaylı bir şekilde açıklanmıştır.

1.3 Yedi Test Prensibi

Son 50 yılda bir dizi test prensibi ortaya atılmıştır ve bunlar, tüm yazılım testleri için ortak genel bir rehber sunmaktadır.

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

Testler, yazılımda hataların mevcut olduğunu gösterebilir, ancak hiç hata kalmadığını ispatlayamaz. Testler, yazılımda

keşfedilmemiş hataların kalma olasılığını azaltır. Yazılımda yeni hatalar bulunamasa bile bu durum yazılımın, kullanıcıların ihtiyaçlarını karşılayacağı anlamına gelmez.

2. Yazılımı %100 test etmek imkansızdır Exhaustive testing is impossible = Kapsamli test imkansizdir.

Yazılımı tüm girdi, ön koşul ve çıktıların kombinasyonları ile test etmek çok basit yazılımlar dışında imkansızdır. Geniş kapsamlı (her durumu kapsayan) testler yapmaya çalışmak yerine, testten daha çok geri dönüş almak için risk analizi, test teknikleri ve öncelikler kullanılmalıdır.

3. Erken test, zaman ve para tasarrufu sağlar

Hataları erken bulmak için, yazılım geliştirme yaşam döngüsünde hem statik hem de dinamik test faaliyetleri mümkün olduğunca erken başlatılmalıdır. Erken teste bazen "sola kaydırma" (shift left) da denir. Testlerin yazılım geliştirme yaşam döngüsünün erken aşamalarında yapılması, sonradan çıkacak yüksek maliyetli değişikliklerin azaltılmasını veya ortadan kaldırılmasını sağlar (bkz. Bölüm 3.1).

4. Hatalar yazılımın belli alanlarında yoğunlaşır Defects cluster together

Yazılımın tüm parçaları(modülleri) göz önüne alınacak olursa, hataların büyük çoğunluğunun bu parçaların küçük bir kısmında bulunacağı görülecektir. Öngörülen hata kümeleri veya test ve canlı kullanım sırasında gözlemlenen hata kümeleri, test aktivitelerini odaklamak için kullanılan bir risk analizi girdisidir (2. prensipte de belirtildiği gibi).

5. Antibiyotik direnci Pesticide paradox

Aynı testler sürekli tekrar edilirse, en sonunda bu testler artık yeni hatalar bulamamaya başlar. Yeni hataları bulmak için mevcut testler ve test verilerinin değiştirilmesi ve yeni testlerin yazılması gerekebilir (Tıpkı sürekli antibiyotik kullanımında bir süre sonra aynı çeşit antibiyotiğin artık bakterileri öldürmede etkili olmaması, bakterilerin direnç kazanması gibi). Yalnız bazı durumlarda örneğin otomatikleştirilmiş regresyon testlerinde antibiyotik direnci prensibi daha önce çalışan yerlerin doğru bir şekilde çalışmaya devam ettiğini göstermek amacıyla faydalı olabilir.

6. Yazılım testi, projenin bağlamına, koşullarına göre değişiklik gösterir Testing is context dependent

Test, farklı proje bağlamlarında farklı şekillerde gerçekleştirilir. Örneğin, güvenlik açısından kritik bir endüstriyel kontrol yazılımı, bir e-ticaret mobil uygulamasından farklı bir şekilde test edilir. Buna başka bir örnek olarak, çevik projelerde testler, sıralı yaşam döngüsü projelerindeki testlerden farklı şekilde yapılır (bkz. Bölüm 2.1).

7. Yeni hata bulamıyoruz başarılı bir yazılım elde ettik yanılgısı

Bazı kurumlar, test uzmanlarının olası tüm testleri uygulamasını ve olası tüm hataları bulmasını bekler, ancak sırasıyla 2. ve 1. ilkeler, bize bunun imkânsız olduğunu söyler. Ayrıca, sadece çok sayıda hatayı bulup çözmenin bir yazılımın başarısını sağlayacağını düşünmek bir yanılgıdır. Örneğin, belirlenen tüm gereksinimleri titizlikle test etmek ve bulunan tüm hataları çözmek, yine de kullanımı zor, kullanıcıların ihtiyaçlarını ve beklentilerini karşılamayan veya diğer rakip yazılımlara kıyasla daha zayıf bir yazılımın üretilmesini engelleyemeyebilir.

Bu ve diğer test prensiplerinin örnekleri için Myers 2011, Kaner 2002 ve Weinberg 2008'e bakınız.

1.4 Test Süreci Test Process

1.4.1 Proje Bağlamında Test Süreci Test Process in Context

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

- Kullanılan yazılım geliştirme yaşam döngüsü modeli ve proje metodolojileri
- Ele alınan test seviyeleri ve test çeşitleri
- Ürün ve proje riskleri
- Kurumun faaliyet alanı
- Aşağıdakileri içeren ancak bunlarla sınırlı olmayan operasyonel kısıtlamalar:
  - Bütçeler ve kaynaklar
  - Süreler

  - Karmaşıklık
  - Sözleşmeden kaynaklanan ve yasal gereksinimler
- Kurumsal politikalar ve uygulamalar
- Sağlanması gereken iç ve dış standartlar

Aşağıdaki bölümlerde organizasyonel test süreçlerinin genel özellikleri açıklanmaktadır:

- Test aktiviteleri ve yapılacak işler
- Test çalışma ürünleri
- Test esası ve test çalışma ürünleri arasındaki izlenebilirlik

Test esasının (ele alınan her test seviyesi veya çeşidi için) ölçülebilir kapsama kriterlerinin belirlenmiş olması çok yararlıdır. Kapsama kriterleri, yazılım test hedeflerine ulaşılmasını sağlayan aktiviteleri yürütmek için anahtar performans göstergeleri (KPI) olarak etkili bir rol oynayabilir (bkz. Bölüm 1.1.1).

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

ISO standardı (ISO/IEC/IEEE 29119-2), test süreçleri hakkında daha fazla bilgi sağlamaktadır.

1.4.2 Test Aktiviteleri ve Yapılacak İşler Test Activities and Tasks

Bir test süreci aşağıdaki ana aktivite gruplarından oluşur:

- Test planlama
- Test gözetimi ve kontrolü
- Test analizi
- Test tasarımı
- Test uyarlama
- Test koşumu
- Test tamamlama

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

Ayrıca, bu aktivite gruplarının birçoğu mantıksal olarak sıralı görünse de, genellikle döngüsel olarak uygulanırlar. Örneğin, çevik yazılım geliştirme, devamlı planlama tarafından desteklenen ve sürekli olarak gerçekleşen yazılım tasarımı, geliştirme ve testlerin küçük döngülerinden oluşur. Dolayısıyla, bu yazılım geliştirme yaklaşımı içinde test aktiviteleri de döngüsel ve sürekli olarak gerçekleşir. Sıralı yazılım geliştirmede bile, kademeli mantıksal aktivite dizisi; kesişme, birleşme, eşzamanlılık veya çıkarmayı içerir. Bu nedenle bu ana aktivitelerin genellikle yazılım ve proje bağlamında düzenlenmesi gerekir.

Test planlama

Test planlaması, test hedeflerini belirleyen aktiviteler ve proje bağlamının dayattığı kısıtlamalar dâhilinde test hedeflerini yerine getirme yaklaşımını içerir (örneğin, uygun test tekniklerini ve işlerini belirlemek ve bunları belirtilen zamanda bitirmek için bir test zaman çizelgesi oluşturmak). Gözetim ve kontrol faaliyetlerinden gelen geri bildirimlere dayanarak test planları gözden geçirilip güncellenebilir. Test planlama bölüm 5.2'de ayrıntılı bir şekilde açıklanmaktadır.

test schedule

Test gözetimi ve kontrolü

Test gözetimi, test planında tanımlanan test gözetim metriklerini kullanarak planlanan ile gerçekleşen ilerlemenin sürekli karşılaştırılmasını içerir. Test kontrolü, test planında belirlenen hedeflere ulaşmak için gerekli önlemlerin alınmasını içerir. Test gözetimi ve kontrolü, çıkış kriterlerinin değerlendirilmesiyle desteklenir; bu kriterler bazı yaşam döngülerinde "Tamamlandı" tanımı olarak da bahsedilir (bkz. ISTQB-AT Temel Seviye Çevik Test Uzmanı Devam Kursu Ders Programı). Örneğin, bir test seviyesinde için test koşumu çıkış kriterlerinin değerlendirilmesi aşağıdakileri içerebilir:

exit criteria

- Test sonuçlarının ve kayıtlarının belirtilen kapsama kriterleriyle karşılaştırılarak kontrol edilmesi

test result log coverage criteria

the level of component or system quality

- Test sonuçlarına ve kayıtlara dayanarak birim veya sistemin kalite seviyesinin değerlendirilmesi
- Daha fazla test gerekip gerekmediğinin belirlenmesi (örneğin, belirli bir ürün riskinin kapsanması için ek testlerin yazılması ve koşturulması)

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

Test gözetimi ve kontrolü bölüm 5.3'te ayrıntılı bir şekilde açıklanmıştır.

Test analizi

to identify testable features — to define associated test conditions — test basis

Test analizi sırasında, test edilebilir özellikleri belirlemek ve ilgili test koşullarını tanımlamak için test esası analiz edilir. Diğer bir deyişle, test analizi ölçülebilir kapsama kriterleri açısından "neyin test edileceğini" belirler.

what to test?

Test analizi aşağıdaki ana aktiviteleri içerir:

- Test seviyesine uygun şekilde test esasını analiz etmek, örneğin:
  - Gereksinimler, fonksiyonel gereksinimler, sistem gereksinimleri, kullanıcı hikâyeleri, epikler, kullanım senaryoları gibi gereksinim spesifikasyonları veya istenen fonksiyonel ve fonksiyonel olmayan birim veya sistem davranışını belirten benzer çalışma ürünleri
  - Sistem veya yazılım mimarisi şemaları veya dokümanları, tasarım spesifikasyonları, çağrı akışları, modelleme şemaları (örneğin, UML veya ER şemaları), arayüz spesifikasyonları gibi tasarım ve uygulama bilgileri veya birim veya sistem yapısını belirten benzer çalışma ürünleri
  - Hayata geçirilmiş kod, veri tabanı meta verileri ve sorgular ve arayüzler
  - Birim veya sistemin fonksiyonel, fonksiyonel olmayan ve yapısal yönlerini dikkate alan risk analizi raporları

UML = Unified Modeling Language

ER = Entity-Relationship

- Aşağıdakiler gibi çeşitli tiplerdeki hataları belirlemek için test esasını ve test öğelerini değerlendirme:
  - Belirsizlikler
  - Çıkarmalar
  - Tutarsızlıklar
  - Yanlışlıklar
  - Çelişkiler
  - Gereksiz ifadeler
- Test edilecek özelliklerin ve özellik gruplarının belirlenmesi
- Test esasının analizine dayanarak fonksiyonel, fonksiyonel olmayan ve yapısal özellikleri, diğer iş ve teknik faktörleri ve risk seviyelerini dikkate alarak her özellik için test koşullarının belirlenmesi ve önceliklendirilmesi
- Test esasının her unsuru ve ilgili test koşulları arasında çift yönlü izlenebilirliğin belirlenmesi (bkz. Bölüm 1.4.3 ve 1.4.4).

Kara kutu, beyaz kutu ve tecrübeye dayalı test tekniklerinin uygulanması, test analizi sürecinde önemli test koşullarının gözden kaçırılma olasılığını azaltmak ve daha kesin ve doğru test koşulları tanımlamak için (bkz. Bölüm 4) faydalı olabilir.

test charter

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

Test analizi sırasında hataların belirlenmesi, özellikle başka bir gözden geçirme tekniğinin kullanılmadığı ve/veya test sürecinin gözden geçirme süreci ile yakından bağlantılı olmadığı durumlarda, önemli bir potansiyel faydadır. Bu tür test analizi faaliyetleri sadece gereksinimlerin tutarlı, doğru ifade edilmiş ve eksiksiz olduğunu doğrulamakla kalmaz, aynı zamanda gereksinimlerin müşteri, kullanıcı ve diğer paydaş ihtiyaçlarını doğru şekilde karşılayıp karşılamadığının sağlamasını da yapar. Örneğin, kodlamadan önce kullanıcı hikâyelerinden ve kabul kriterlerinden test koşulları ve test senaryoları oluşturmayı içeren, davranış odaklı yazılım geliştirme (BDD) ve kabul testi odaklı yazılım geliştirme (ATDD) gibi tekniklerle kullanıcı hikâyeleri ve kabul kriterleri doğrulanır, sağlaması yapılır ve hataları bulunur (bkz. ISTQB Temel Seviye Çevik Test Uzmanı Devam Kursu Ders Programı).

BDD => behavior driven development = davranış odaklı geliştirme

ATDD => acceptance test driven development = kabul testi odaklı geliştirme

Test tasarımı

Test tasarımı sırasında test koşulları; üst seviye test senaryoları, üst seviye test senaryo grupları ve test yazılımları gibi ayrıntılar belirlenir. Bu nedenle, test analizi "ne test edilecek?" sorusuna cevap verirken, test tasarımı "nasıl test edilecek?" sorusuna cevap verir.

what to test?

how to test?

Test tasarımı aşağıdaki ana aktiviteleri içerir:

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

Test koşullarının test senaryolarına ve test senaryosu gruplarına genişletilmesinde genellikle test teknikleri kullanılır (bkz. Bölüm 4).

Test analizinde olduğu gibi test tasarımı da test esasındaki benzer hata çeşitlerinin belirlenmesini sağlayabilir. Yine test analizinde olduğu gibi, test tasarımı sırasında hataların belirlenmesi önemli bir potansiyel faydadır.

Test uyarlama

Test uyarlama sırasında, test koşumu için eğer gerekiyorsa test yazılımı oluşturulur ve/veya tamamlanır; buna test senaryolarından test prosedürleri üretmek de dâhildir. Dolayısıyla, test tasarımı "nasıl test edilecek?" sorusuna cevap verirken, test uyarlama "şu anda testleri koşmak için gerekli şeylere sahip miyiz?" sorusuna cevap arar.

Test uyarlama aşağıdaki ana aktiviteleri içerir:

do we now have everything in place to run the tests?

- Test prosedürlerinin yazılması, önceliklendirilmesi ve gerekiyorsa test betiklerinin otomasyonu
- Test prosedürlerinden ve yazıldıysa otomatikleştirilmiş test betiklerinden test grupları oluşturulması
- Test gruplarını, test senaryolarını ve otomatikleştirilmiş test betiklerini içeren test koşum çizelgesi oluşturulması (bkz. Bölüm 5.2.4).
- Test ortamının oluşturulması (gerekiyorsa test kuluçkası, servis sanallaştırması, simülatörler ve diğer altyapı öğelerini dâhil ederek) ve ihtiyaç duyulan her şeyin doğru şekilde kurulduğunun doğrulanması
- Test verilerinin hazırlanması ve test ortamına düzgün şekilde yüklenmesinin sağlanması
- Test esası, test koşulları, test senaryoları, test prosedürleri ve test grupları arasında çift yönlü izlenebilirliğin doğrulanması ve güncellenmesi (bkz. Bölüm 1.4.4)

Test tasarımı ve test uyarlama görevleri genellikle birlikte hayata geçirilir.

Keşif testlerinde ve diğer tecrübeye dayalı test çeşitlerinde, test tasarımı ve uyarlaması, test koşumunun bir parçası olarak gerçekleştirilebilir ve dokümante edilebilir. Keşif testleri, test tüzüklerine (test analizinin bir parçası olarak üretilir) dayandırılmış olabilir ve keşif testleri tasarlandıktan ve uyarlandıktan sonra hemen koşturulur (bkz. Bölüm 4.4.2).

Test koşumu

Test koşumu sırasında, test grupları test koşum çizelgesine uygun olarak çalıştırılır. Test koşumu aşağıdaki ana aktiviteleri içerir:

- Test öğesinin/öğelerinin veya test nesnesinin, test aracının/araçlarının ve test yazılımının kimlik numaralarının ve versiyonlarının kaydedilmesi
- Testlerin manuel olarak veya test koşum araçları kullanılarak koşulması
- Gerçekleşen sonuçlarla beklenen sonuçların karşılaştırılması
- Olası nedenlerin belirlenmesi için anomalilerin analiz edilmesi (örneğin, koddaki hatalar nedeniyle arızalar oluşabilir, ancak yanlış pozitifler de ortaya çıkabilir [bakınız bölüm 1.2.3])
- Gözlemlenen arızalara dayanılarak hataların bildirilmesi (bkz. Bölüm 5.6)

- Test koşum sonucunun kaydedilmesi (örneğin, başarılı, başarısız, engellendi) (pass, fail, blocked)
- Bir anomali için ya da planlı testlerin bir parçası olarak test aktivitelerinin tekrarlanması (örneğin, düzeltilmiş bir testin, onaylama testinin ve/veya regresyon testlerinin koşturulması)
- Test esası, test koşulları, test senaryoları, test prosedürleri ve test sonuçları arasındaki çift yönlü izlenebilirliğin doğrulanması ve güncellenmesi.

Test tamamlama

Test tamamlama aktiviteleri; test projesinde elde edilen deneyimi, proje boyunca üretilen test yazılımını ve elde edilen diğer ilgili bilgileri toplar ve bir araya getirir. Test tamamlama aktiviteleri, bir yazılımın piyasaya sürülmesi, bir test projesinin tamamlanması (veya iptal edilmesi), bir çevik proje döngüsünün tamamlanması (örneğin bir geriye dönük değerlendirme toplantısının parçası olarak), bir test seviyesinin tamamlanması veya bir bakım sürümünün tamamlanması gibi proje (milestone) kilometre taşlarında gerçekleştirilir.

Test tamamlama aşağıdaki ana aktiviteleri içerir:

- Tüm hata raporlarının kapatılıp kapatılmadığının kontrol edilmesi, test koşumu sonunda çözülmeden kalan hatalar için değişiklik taleplerinin ürün iş listesine girilmesi
- Paydaşlara iletilmek üzere bir test özet raporunun oluşturulması
- Test ortamının, test verilerinin, test altyapısının ve diğer test yazılımlarının daha sonra tekrar kullanılmak üzere sonlandırılması ve arşivlenmesi
- Test yazılımının bakım ekiplerine, diğer proje ekiplerine ve/veya test yazılımının kullanımından faydalanabilecek diğer paydaşlara teslim edilmesi
- Gelecek döngüler, sürümler ve projeler için gereken değişikliklerin belirlenmesi amacıyla tamamlanan test faaliyetlerinden çıkarılan derslerin analiz edilmesi
- Test süreci olgunluğunun artırılması için toplanan bilgilerin kullanılması

1.4.3 Test Çalışma Ürünleri Test Work Products

Test çalışma ürünleri, test sürecinin bir parçası olarak oluşturulur. Kurumların test sürecini uygulama şekillerinde önemli farklılıklar olduğu gibi, bu süreçte oluşturulan çalışma ürünleri çeşitlerinde, bu çalışma ürünlerinin düzenlenme ve yönetilme şekillerinde ve bu çalışma ürünleri için kullanılan isimlerde de önemli farklılıklar olabilir. Bu ders programında yukarıda belirtilen test sürecine ve bu ders programında ve ISTQB terimler sözlüğünde açıklanan çalışma ürünlerine bağlı kalınmıştır. ISO standardı (ISO/IEC/IEEE 29119-3), test çalışmaları ürünleri için bir rehber görevi de görebilir.

Bu bölümde açıklanan test çalışma ürünlerinin çoğu, test yönetimi araçları ve hata yönetimi araçları kullanılarak kaydedilebilir ve yönetilebilir (bkz. Bölüm 6).

Test planlama çalışma ürünleri

Test planlama çalışma ürünleri genellikle bir veya daha fazla test planı içerir. Test planı, diğer test çalışma ürünlerinin izlenebilirlik bilgileri sayesinde ilişkili olacağı test esası hakkında bilgiler (aşağıya ve bölüm 1.4.4'e bakınız) ve test gözetimi ve kontrolü sırasında kullanılacak çıkış kriterlerini (veya Tamamlandı tanımını) içerir. Test planları bölüm 5.2'de açıklanmaktadır.

Test gözetimi ve kontrolü çalışma ürünleri

Test gözetimi ve kontrolü çalışma ürünleri genellikle, test ilerleme raporları (sürekli ve/veya düzenli olarak üretilir) ve test özet raporları (çeşitli kilometre taşlarında üretilir) dâhil olmak üzere birçok test raporu çeşitlerini içerir. Tüm test raporları, test koşum sonuçlarının özetlenmesi de dâhil olmak üzere, rapor tarihi itibariyle test ilerlemesi hakkında ilgili paydaşları ilgilendiren ayrıntıları sağlamalıdır.

Test gözetimi ve kontrolü çalışma ürünleri ayrıca görev tamamlama, kaynak tahsisi ve kullanımı ve efor gibi proje yönetimi konularını da ele almalıdır.

Test gözetimi, kontrolü ve bu aktiviteler sırasında oluşturulan çalışma ürünleri, bu ders programında Bölüm 5.3'te daha detaylı olarak açıklanmıştır.

Test analizi çalışma ürünleri

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

Test tasarımı çalışma ürünleri

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

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

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

Test uyarlama çalışma ürünleri

Test uyarlama çalışma ürünleri aşağıdakileri içerir:

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

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

Bazı durumlarda, test uyarlamada, servis sanallaştırma ve otomatize test betikleri gibi araçları kullanan veya onlar tarafından kullanılan çalışma ürünleri oluşturulur.

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

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

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

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

!!! Burada normalde Test execution (Test koşum) ve Test completion (Test tamamlama) work products konuları yer aliyordu. Bu eksik konular icin İngilizce müfredatta sayfa 24'e bakabilirsiniz!!!

2 örn.

1.4.4 Test Esası ve Test Çalışma Ürünleri Arasında İzlenebilirlik

Traceability between the Test Basis and Test Work Products

Bölüm 1.4.3'te belirtildiği gibi, test çalışma ürünleri ve bu çalışma ürünlerinin isimleri önemli ölçüde değişmektedir. Bu değişikliklerden bağımsız olarak, etkin test gözetimi ve kontrolünü uygulamak için, yukarıda anlatıldığı gibi, test süreci boyunca test esasının her unsuru ile bu unsurla ilişkili çeşitli test çalışma ürünleri arasında izlenebilirliğin oluşturulması ve sürdürülmesi önemlidir. Test kapsamının değerlendirilmesine ek olarak, iyi bir izlenebilirlik aşağıdakileri de destekler:

- Değişikliklerin etkisinin analiz edilmesi
- Testlerin denetlenebilir şekilde yapılması
- BT yönetişim kriterlerinin karşılanması
- Test esasındaki unsurların durumunu (ör. testleri geçen gereksinimler, testlerde başarısız olan gereksinimler ve bekleyen testleri olan gereksinimler) içerecek şekilde test ilerleme raporlarının ve test özet raporlarının anlaşılabilirliğinin artırılması

- Testlerin teknik yönlerinin, anlaşılır şekilde paydaşlara aktarılması
- İş hedeflerine göre ürün kalitesinin, süreç kapasitesinin ve proje ilerlemesinin değerlendirilmesi için bilgi sağlanması

Bazı test yönetim araçları, bu bölümde belirtilen test çalışma ürünlerinin bir kısmı veya tamamı ile eşleşen test çalışma ürünleri sağlar. Bazı kurumlar çalışma ürünlerini düzenlemek ve ihtiyaç duydukları bilgi izlenebilirliğini sağlamak için kendi yönetim sistemlerini geliştirirler.

1.5 Test Etme Psikolojisi

1.5.1 İnsan Psikolojisi ve Test

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

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

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

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

Tipik test hedefleri önceki bölümlerde tartışılmıştır (bkz. Bölüm 1.1). Doğru test hedeflerinin açıkça tanımlanmasının önemli psikolojik etkileri vardır. Çoğu kişi, planlarını ve davranışlarını ekip, yönetim ve diğer paydaşlar tarafından belirlenen hedeflerle uyumlu hale getirme eğilimindedir. Test uzmanlarının bu hedeflere asgari kişisel önyargılarla uyması da önemlidir.

1.5.2 Test Uzmanlarının ve Yazılımcıların Düşünce Tarzları

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

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

curiosity, professional pessimism, a critical eye, attention to detail, and a motivation for good and positive communications and relationships

Bir yazılımcının düşünce tarzı, bir test uzmanının düşünce tarzının bazı unsurlarını içerebilir, ancak yazılımcılar, genellikle

— çözümler tasarlamak ve oluşturmakla ilgilenir, bu çözümlerde neyin yanlış olabileceğini fazla düşünmez. Ek olarak, psikolojideki doğrulama sapması da kendi çalışmalarındaki hataları bulmalarını zorlaştırır.

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

2. Yazılım Geliştirme Yaşam Döngüsü Boyunca Test

Testing Throughout the Software Development Lifecycle

100 dakika

Anahtar kelimeler

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

Yazılım Geliştirme Yaşam Döngüsü Boyunca Test için Öğrenme Hedefleri

Learning Objectives

2.1 Yazılım Geliştirme Yaşam Döngüsü Modelleri

Software Development Lifecycle Models

FL-2.1.1 (K2) Yazılım geliştirme yaşam döngüsü içindeki yazılım geliştirme aktiviteleri ile test aktiviteleri arasındaki ilişkileri açıklayın

FL-2.1.2 (K1) Yazılım geliştirme yaşam döngüsü modellerinin neden proje ve ürün özellikleri bağlamına uyarlanması gerektiğini açıklayın

2.2 Test Seviyeleri

Test Levels

FL-2.2.1 (K2) Farklı test seviyelerini hedefler, test esası, test nesneleri, genel hatalar ve arızalar ile yaklaşımlar ve sorumluluklar açısından karşılaştırın.

2.3 Test Çeşitleri

Test Types

FL-2.3.1 (K2) Fonksiyonel, fonksiyonel olmayan ve beyaz kutu testlerini karşılaştırın

FL-2.3.2 (K1) Fonksiyon, fonksiyonel olmayan ve beyaz kutu testlerinin her test seviyesinde gerçekleştiğinin farkında olun

FL-2.3.3 (K2) Onaylama testleri ve regresyon testlerinin amaçlarını karşılaştırın

2.4 Bakım Testleri

Maintenance Testing

FL-2.4.1 (K2) Bakım testleri için tetikleyicileri özetleyin

FL-2.4.2 (K2) Bakım testlerinde etki analizinin rolünü açıklayın

2.1 Yazılım Geliştirme Yaşam Döngüsü Modelleri

Bir yazılım geliştirme yaşam döngüsü modeli, yazılım geliştirme projesinin her aşamasında gerçekleştirilen aktivite çeşitlerini ve aktivitelerin birbirleriyle mantıksal ve kronolojik olarak nasıl ilişkili olduğunu açıklar. Her biri farklı test yaklaşımları gerektiren çok sayıda farklı yazılım geliştirme yaşam döngüsü modeli vardır.

2.1.1 Yazılım Geliştirme ve Yazılım Testi — Software Development and Software Testing

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

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

- Her yazılım geliştirme aktivitesine karşılık gelen bir test aktivitesi vardır.
- Her test seviyesinin bu seviyeye özgü test hedefleri vardır.
- Belirli bir test seviyesi için test analizi ve tasarımı, ilgili yazılım geliştirme faaliyeti sırasında başlar.
- Test uzmanları, gereksinimleri ve tasarımı belirlemek ve iyileştirmek için tartışmalara katılır ve taslaklar hazırlanır hazırlanmaz çalışma ürünlerinin (örneğin; gereksinimler, tasarım, kullanıcı hikâyeleri vb.) gözden geçirilmesinde yer alır.

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

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

- Sıralı yazılım geliştirme modelleri — Sequential development models
- Döngüsel ve artımlı yazılım geliştirme modelleri — Iterative and incremental development models

Sıralı bir yazılım geliştirme modeli, yazılım geliştirme sürecini doğrusal, sıralı bir faaliyetler akışı olarak tanımlar. Bu, yazılım geliştirme sürecinde her aşamanın önceki aşama tamamlandığında başlaması gerektiği anlamına gelir. Teorik olarak aşamaların kesişimi yoktur, ancak uygulamada takip eden aşamadan erken geri bildirim almak faydalıdır.

Waterfall model

Şelale modelinde, yazılım geliştirme faaliyetleri (örneğin; gereksinim analizi, tasarım, kodlama, testler) birbiri ardına tamamlanır. Bu modelde, test aktiviteleri ancak diğer tüm yazılım geliştirme aktiviteleri tamamlandıktan sonra gerçekleştirilir.

V-model

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

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

incremental

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

iterative

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

Bazı örnekleri şunlardır:

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

- Spiral (veya prototipleme): Görsel veya deneysel yazılım özellikleri oluşturmayı içerir, bunlardan bazıları ilerleyen aşamalarda büyük oranda güncellenebilir veya bunlardan vazgeçilebilir.

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

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

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

Çevik yazılım geliştirmede test yaklaşımları hakkında daha fazla bilgi için, bkz. ISTQB-AT Temel Seviye Çevik Test Uzmanı Devam Kursu Ders Programı, Black 2017, Crispin 2008 ve Gregory 2015.

in Context

2.1.2 Proje Bağlamında Yazılım Geliştirme Yaşam Döngüsü Modelleri

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

Projenin bağlamına bağlı olarak, test seviyelerinin ve/veya test aktivitelerinin birleştirilmesi veya yeniden düzenlenmesi gerekebilir. Örneğin, ticari paket yazılımın (COTS) daha büyük bir sisteme entegrasyonu için, birlikte çalışabilirlik testleri sistem entegrasyonu testi seviyesinde (örneğin; altyapıya ve diğer sistemlere entegrasyon) ve kabul testi seviyesinde (kullanıcı kabul testleri ve operasyonel kabul testleri ile birlikte fonksiyonel ve fonksiyonel olmayan testler) yapılabilir.
Test seviyelerinin tanımları için bölüm 2.2'ye ve test çeşitlerinin tanımları için bölüm 2.3'e bakınız.

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

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

2.2 Test Seviyeleri — Test Levels

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

- Birim testleri — Component testing
- Entegrasyon testleri — Integration testing
- Sistem testleri — System testing
- Kabul testleri — Acceptance testing

Test seviyelerinin karakteristik özellikleri aşağıda verilmiştir:

- Spesifik hedefler

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

Her test seviyesi için uygun bir test ortamı gereklidir. Örneğin kabul testinde canlı ortam benzeri bir test ortamı ideal iken, birim testinde yazılımcılar genellikle kendi yazılım geliştirme ortamlarını kullanırlar.

2.2.1 Birim Testi

Component testing

Component testing

Birim testinin hedefleri

unit or module testing

Birim testleri (bileşen veya modül testleri olarak da bilinir), ayrı olarak test edilebilen bağımsız birimlere odaklanır. Birim testinin hedefleri aşağıda verilmiştir:

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

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

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

Test esası

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

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

Test nesneleri

Birim testlerinde tipik test nesneleri aşağıda verilmiştir:

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

Tipik hatalar ve arızalar

Birim testlerindeki yaygın hata ve arıza örnekleri aşağıda verilmiştir:

- Yanlış fonksiyonalite (örneğin, tasarım gereksinimlerinde açıklanandan farklı)
- Veri akışı problemleri

- Hatalı kod ve mantık

Hatalar, genellikle tanımlı bir hata yönetim süreci olmadan, bulundukları anda giderilir. Bununla birlikte, yazılımcılar hataları raporlarsa, kök neden analizi ve süreç iyileştirmesi için önemli bilgiler sağlanmış olur.

Birim testlerine özgü yaklaşımlar ve sorumluluklar

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

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

2.2.2 Entegrasyon Testi

Entegrasyon testinin hedefleri

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

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

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

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

- Birim entegrasyonu testleri, entegre birimler arasındaki etkileşimlere ve arayüzlere odaklanır. Birim entegrasyonu testleri, birim testlerinden sonra gerçekleştirilir ve genellikle otomatikleştirilmiştir. Döngüsel ve artımlı yazılım geliştirmede birim entegrasyonu testleri genellikle sürekli entegrasyon sürecinin bir parçasıdır.
- Sistem entegrasyonu testleri, sistemler, paketler ve mikroservisler arasındaki etkileşimlere ve arayüzlere odaklanır.

Sistem entegrasyonu testleri, harici kurumlar (örneğin ağ servisleri) ile etkileşimleri ve onlar tarafından sağlanan arayüzleri de kapsayabilir. Bu durumda, yazılımı geliştirmekte olan kurumun harici arayüzler üzerinde bir kontrolü olmadığı için testlerde çeşitli zorluklar yaşanabilir (örneğin, harici kurumun kodundaki testi engelleyen hataların giderilmesi, test ortamı düzenlemeleri vb.). Sistem entegrasyonu testleri, sistem testlerinden sonra veya devam eden sistem test faaliyetlerine paralel olarak yapılabilir (hem sıralı yazılım geliştirmede hem de döngüsel ve artımlı yazılım geliştirmede).

Test esası

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

- Yazılım ve sistem tasarımı
- Sekans (dizi) diyagramı
- Arayüz ve iletişim protokolü gereksinimleri
- Kullanım senaryoları

- Birim veya sistem seviyesindeki mimari
- İş akışları
- Harici arayüz tanımları

Test nesneleri

Entegrasyon testlerinde ele alınan tipik test nesneleri aşağıda verilmiştir:

- Alt-sistemler
- Veritabanları
- Altyapı
- Arayüzler
- API'ler
- Mikroservisler

Tipik hatalar ve arızalar

Birim entegrasyon testlerindeki tipik hata ve arıza örnekleri aşağıda verilmiştir:

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

Sistem entegrasyon testlerindeki tipik hata ve arıza örnekleri aşağıda verilmiştir:

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

Entegrasyon testlerine özgü yaklaşımlar ve sorumluluklar

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

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

Entegrasyon testleri ve entegrasyon stratejisi birimler veya sistemler oluşturulmadan önce planlanırsa, bu birimler veya sistemler, testin en verimli şekilde gerçekleştirilmesi için gereken sıraya göre oluşturulabilir. Sistematik entegrasyon stratejileri, sistem mimarisine (örneğin yukarıdan aşağıya veya aşağıdan yukarıya), fonksiyonel görevlere, işlem dizilerine veya

sistemin veya birimlerin başka bir sıralamasına dayanabilir. Hata bulma sürecini basitleştirmek ve hataları erkenden bulmak için entegrasyon "büyük patlama" (tüm birimleri veya sistemleri tek bir seferde birleştirmek) şeklinde değil, artımlı (bir seferinde az sayıda ek birim veya sistem) bir şekilde olmalıdır. En karmaşık arayüzlerin risk analizi, entegrasyon testlerini doğru alanlara odaklamaya yardımcı olabilir.

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

2.2.3 Sistem Testi

Sistem testinin hedefleri

Sistem testleri, bütün bir sistemin veya ürünün davranış ve yeteneklerine odaklanır ve genellikle sistemin gerçekleştirebileceği uçtan uca işleri (fonksiyonalite) ve bu işleri gerçekleştirirken gösterdiği fonksiyonel olmayan davranışları ele alır. Sistem testlerinin hedefleri aşağıda verilmiştir:

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

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

Sistem test ortamı ideal olarak canlı ortama karşılık gelmelidir.

Test esası

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

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

Test nesneleri

Sistem testlerinde ele alınan tipik test nesneleri aşağıda listelenmiştir:

- Uygulamalar
- Donanım/yazılım sistemleri
- İşletim sistemleri

- Test edilen sistem (SUT
- Sistem yapılandırması ve yapılandırma verileri

Tipik hatalar ve arızalar

Sistem testlerindeki tipik hata ve arıza örnekleri aşağıda listelenmiştir:

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

Spesifik yaklaşımlar ve sorumluluklar

Sistem testleri, sistemin bir bütün olarak uçtan uca (fonksiyonel ve fonksiyonel olmayan) davranışına odaklanmalıdır. Sistem testleri, test edilecek sisteme özgü en uygun teknikleri kullanmalıdır (bkz. Bölüm 4). Örneğin, fonksiyonel özelliklerin gereksinimlerde tanımlandığı şekilde olup olmadığını doğrulamak için bir karar tablosu oluşturulabilir.
Sistem testlerini genellikle bağımsız test uzmanları gerçekleştirir. Gereksinimlerdeki hatalar (örneğin; eksik kullanıcı hikâyeleri, yanlış belirlenen kullanıcı gereksinimleri vb.) beklenen sistem davranışının anlaşılamamasına veya yanlış anlaşılmasına yol açabilir. Bu tür durumlar, yanlış pozitif ve yanlış negatiflere neden olabilir; bu da, zaman kaybına ve hata bulma etkinliğinin azalmasına neden olur. Test uzmanlarının kullanıcı hikâyesinin geliştirilmesi veya gözden geçirilmesi gibi statik test aktivitelerine erken aşamalarda katılmaları bu gibi durumların görülme sıklığını azaltır.

2.2.4 Kabul Testi Acceptance Testing

Kabul testinin hedefleri

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

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

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

Yaygın kullanılan kabul testi çeşitleri arasında aşağıdakiler yer alır:

- Kullanıcı kabul testleri
- Operasyonel kabul testleri
- Sözleşmeye dayalı ve yasal kabul testleri
- Alfa ve beta testleri

Bunları her biri aşağıdaki dört alt bölümde açıklanmaktadır.

Kullanıcı kabul testleri (KKT) User acceptance testing (UAT)

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

2 Operasyonel kabul testleri (OKT) Operational acceptance testing (OAT)

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

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

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

3 Sözleşmeye dayalı ve yasal kabul testleri OperatiContractual and regulatory acceptance testing

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

Yasal kabul testleri, hükümet, yasal veya emniyet düzenlemeleri gibi uyulması gereken düzenlemelere göre yapılır. Yasal kabul testleri genellikle kullanıcılar veya bağımsız test uzmanları tarafından gerçekleştirilir, bazen yasal kurumların sonuçlara şahitlik etmesi veya sonuçları denetlemesi gerekebilir.

Sözleşmeye dayalı ve yasal kabul testlerinin temel amacı, sözleşmeye veya yasal düzenlemelere uyumluluk sağlandığına dair güven oluşturmaktır.

4 Alfa ve beta testleri

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

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

Test esası

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

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

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

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

Tipik test nesneleri

Herhangi bir kabul testinde ele alınabilecek tipik test nesneleri aşağıda verilmiştir:

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

Tipik hatalar ve arızalar

Herhangi bir kabul testi için tipik hata örnekleri aşağıda verilmiştir:

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

Kabul testine özgü yaklaşımlar ve sorumluluklar

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

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

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

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

2.3 Test Çeşitleri Test Types

Test çeşidi, belirlenmiş test hedeflerine dayanarak bir yazılımın belirli özelliklerini veya bir sistemin bir bölümünü test etmeyi amaçlayan bir test aktiviteleri grubudur. Bu test hedefleri aşağıdakileri içerebilir:

- Bütünlük, doğruluk ve uygunluk gibi fonksiyonel kalite özelliklerinin değerlendirilmesi
- Güvenilirlik, performans, güvenlik, uyumluluk ve kullanılabilirlik gibi fonksiyonel olmayan kalite özelliklerinin değerlendirilmesi
- Birim veya sistemin yapısının veya mimarisinin doğru, eksiksiz ve gereksinimlerde belirtildiği gibi olup olmadığının değerlendirilmesi
- Değişikliklerin etkilerinin değerlendirilmesi: hataların giderildiğini onaylama (onaylama testleri) ve düzeltilen hatanın veya yapılan değişikliğin istenmeyen değişiklikleri tetikleyip tetiklemediğini bulma (regresyon testleri) gibi.

2.3.1 Fonksiyonel Testler

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

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

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

Fonksiyonel testlerin bütünlük derecesi, fonksiyonel kapsam ile ölçülebilir. Fonksiyonel kapsam, fonksiyonel gereksinimin testlerle ne kadar ele alındığını belirtir ve gereksinimin yüzdesi olarak ifade edilir. Testler ve fonksiyonel gereksinimler arasındaki izlenebilirlik kullanılarak bu gereksinimlerin testlerde ele alınan yüzdesi hesaplanabilir ve böylece potansiyel olarak kapsam eksiklikleri belirlenebilir.
Fonksiyonel test tasarımı ve koşumu, yazılımın çözdüğü belirli bir iş probleminin bilgileri (örneğin, petrol ve gaz endüstrileri için jeolojik modelleme yazılımı) veya yazılımın hizmet ettiği özel roller (örneğin, etkileşimli eğlence sağlayan bilgisayar oyun yazılımı) gibi özel bilgi veya becerileri içerebilir.

2.3.2 Fonksiyonel Olmayan Testler

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

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

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

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

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

Fonksiyonel olmayan kalite özelliklerinin testleriyle ilgili daha fazla bilgi için ISTQB-ATA İleri Seviye Test Analisti Ders Programı, ISTQB-ATTA İleri Seviye Teknik Test Analisti Ders Programı, ISTQB-SEC İleri Seviye Güvenlik Test Uzmanı Ders Programı ve diğer ISTQB uzman modüllerine bakınız.

2.3.3 Beyaz Kutu Testleri

Beyaz kutu testleri, sistemin iç yapısına dayanan testleri oluşturur. İç yapı; kod, mimari, iş akışları ve/veya sistem içindeki veri akışlarını içerebilir (bkz. Bölüm 4.3).

Beyaz kutu testlerinin bütünlük derecesi yapısal kapsam ile ölçülebilir. Yapısal kapsam, bir yapısal öğenin testlerle ne ölçüde ele alınmış olduğunu belirtir ve kapsanan öğenin yüzdesi olarak ifade edilir.

Birim testi seviyesinde, kod kapsamı test edilen birim kodunun yüzdesine dayanır ve birimde test edilen komutların yüzdesi veya test edilen kararların yüzdesi gibi kodun farklı yönleri (kapsam öğeleri) açısından ölçülebilir. Bu kapsam çeşitlerine toplu olarak kod kapsamı denir. Birim entegrasyon testleri seviyesinde beyaz kutu testleri, birimler arasındaki arayüzler gibi sistem mimarisine dayanabilir ve yapısal kapsam, testler ile ele alınmış arayüzlerin yüzdesi olarak ölçülebilir.
Beyaz kutu test tasarımı ve koşumu, kodun yazılma şekli (örneğin kod kapsamı araçlarını kullanmak), verilerin nasıl depolandığı (örneğin olası veritabanı sorgularını değerlendirmek), kapsam araçlarının nasıl kullanılacağı ve sonuçlarının doğru şekilde yorumlanması gibi özel bilgi veya becerileri içerebilir.

2.3.4 Değişiklikle İlgili Testler Change-related Testing

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

Confirmation testing

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

Regression testing

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

Onaylama testleri ve regresyon testleri tüm test seviyelerinde yapılabilir.

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

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

2.3.5 Test Çeşitleri ve Test Seviyeleri Test Types and Test Levels

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

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

Aşağıdakiler fonksiyonel olmayan testlerin örnekleridir:

- Birim testleri için, karmaşık bir faiz hesaplaması yapmak için gereken işlemci (CPU) işlem sayısı test edilebilir.

- Birim entegrasyon testleri için, kullanıcı arayüzünden iş mantığına iletilen verilerden kaynaklanan arabellek aşımı güvenlik açıkları test edilebilir.
- Sistem testleri için, sunum katmanının tüm desteklenen tarayıcılarda ve mobil cihazlarda çalışıp çalışmadığı taşınabilirlik testleri yapılarak test edilebilir.
- Sistem entegrasyon testleri için, kredi puanı mikroservisinin yanıt vermemesi durumunda, sistem sağlamlığını değerlendirmek üzere güvenilirlik testleri yapılabilir.
- Kabul testleri için, bankacının kredi işleme arayüzünün engelliler tarafından erişilebilirliğini değerlendirmek üzere kullanılabilirlik testleri yapılabilir.

Aşağıdakiler beyaz kutu testlerinin örnekleridir:

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

Son olarak, aşağıdakiler değişiklikle ilgili testler için örneklerdir:

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

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

2.4 Bakım Testleri Maintenance Testing

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

Bakımın bir parçası olarak herhangi bir değişiklik yapıldığında, hem değişikliklerin ne kadar başarılı olduğunu değerlendirmek hem de sistemin değişmeyen kısımlarındaki (genellikle sistemin büyük kısmıdır) olası yan etkileri (ör. regresyonlar) kontrol etmek için bakım testleri yapılmalıdır.
Bakım testleri, sistemde yapılan değişiklikleri ve değişikliklerden etkilenmiş olabilecek değişmeyen parçaları test etmeye odaklanır. Bakım, planlanan ve planlanmamış sürümleri (düzeltmeler) içerebilir.

Bir bakım sürümü, kapsamına bağlı olarak farklı test çeşitlerini kullanarak birçok test seviyesinde bakım testleri gerektirebilir. Bakım testlerinin kapsamı aşağıdakilere bağlıdır:

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

2.4.1 Bakım için Tetikleyiciler Triggers for Maintenance

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

Bakım için tetikleyicileri aşağıdaki gibi sınıflandırabiliriz:

Modification
- Değişiklik: planlanan iyileştirmeler (örneğin sürüm bazlı), düzeltici ve acil durum değişiklikleri, operasyonel ortamdaki değişiklikler (planlanan işletim sistemi veya veritabanı yükseltmeleri gibi), ticari paket yazılımın (COTS) üst bir sürüme yükseltilmesi, hatalar ve güvenlik açıkları için yamalar gibi.

Migration
- Taşıma: yeni ortamın ve değiştirilen yazılımın operasyonel testlerini gerektirebilen, bir platformdan diğerine taşıma gibi veya başka bir uygulamadan gelen verilerin dönüştürülme ihtiyacı sonucu testlerinin yapılması gibi.

Retirement
- Kullanımdan kaldırma: bir uygulamanın kullanım ömrünün sonuna gelinmesi gibi.

Bir uygulama veya sistemin kullanımı sonlandırılırken, verilerin uzun süre saklanması gerekliyse, veri taşıma testleri veya arşivleme gerekebilir. Uzun saklama süreleri için veriler arşivlendikten sonra geri yükleme/geri alma prosedürlerinin test edilmesi de gerekebilir. Ek olarak, kullanımda kalan herhangi bir fonksiyonalitenin hala çalıştığından emin olmak için regresyon testleri gerekebilir.

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

2.4.2 Bakım için Etki Analizi Impact Analysis for Maintenance

Etki analizi, bir değişikliğin beklenen ve olası yan etkilerini, istenilen sonuçların elde edilip edilemeyeceğini ve değişiklikten etkilenecek alanları belirlemek amacıyla yapılır. Etki analizi bir değişikliğin mevcut testler üzerindeki etkisinin belirlenmesinde de yardımcı olabilir. Sistemdeki yan etkilerin ve etkilenen alanların regresyon testleri gerekli değişiklikler yapıldıktan sonra yapılmalıdır.
Bir değişiklik yapılmadan önce, sistemin diğer alanlarında yaratacağı potansiyel sonuçlara dayanarak değişikliğin yapılıp yapılmayacağına karar vermek için etki analizi yapılabilir.

Aşağıdaki durumlarda etki analizi zor olabilir:

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

3 Statik Testler — 135 dakika

Anahtar kelimeler

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

Statik Testler için Öğrenme Hedefleri:

3.1 Statik Testlerin Temelleri — Static Testing Basics

FL-3.1.1 (K1) Farklı statik test teknikleri ile incelenebilecek çalışma ürünü çeşitlerini ayırt edin.

FL-3.1.2 (K2) Statik testlerin önemini tanımlamak için örnekler verin.

FL-3.1.3 (K2) Hedeflerini, belirlenecek hata çeşitlerini ve bu tekniklerin yazılım yaşam döngüsü içindeki rolünü göz önünde bulundurarak statik ve dinamik teknikler arasındaki farkları açıklayın.

3.2 Gözden Geçirme Süreci — Review Process

FL-3.2.1 (K2) Çalışma ürünü gözden geçirme sürecinin faaliyetlerini özetleyin.

FL-3.2.2 (K1) Resmi bir gözden geçirmede farklı rol ve sorumlulukları tanımlayın.

FL-3.2.3 (K2) Farklı gözden geçirme çeşitleri arasındaki farkları açıklayın: resmi gözden geçirme, üzerinden geçme, teknik gözden geçirme ve teftiş — informal review, walkthrough, technical review, inspection

FL-3.2.4 (K3) Hataları belirlemek için bir çalışma ürününe gözden geçirme tekniği uygulayın.

FL-3.2.5 (K2) Başarılı bir gözden geçirmeye katkıda bulunan faktörleri açıklayın.

3.1 Statik Testin Temelleri

Test edilen yazılımın çalıştırılmasını gerektiren dinamik testlerin aksine statik testler, yazılımın veya diğer çalışma ürünlerinin manuel incelenmesine (gözden geçirmelere) veya kodun veya diğer çalışma ürünlerinin araç kullanılarak değerlendirilmesine (statik analizlere) dayanır. Her iki statik test çeşidi de test edilen kod veya çalışma ürününü gerçekten çalıştırmadan değerlendirir.

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

3.1.1 Statik Teste Dahil Edilebilecek Çalışma Ürünleri

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

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

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

3.1.2 Statik Testin Faydaları Benefits of Static Testing

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

- Hataların daha verimli bir şekilde, dinamik testlerin yürütülmesinden önce bulunması ve düzeltilmesi
- Dinamik testlerle kolayca bulunamayan hataların bulunması
- Gereksinimlerdeki tutarsızlıkları, belirsizlikleri, çelişkileri, çıkarmaları, yanlışlıkları ve fazlalıkları ortaya çıkararak tasarımdaki veya kodlamadaki hataların önlenmesi
- Yazılım geliştirme verimliliğinin artırılması (ör. gelişmiş tasarım ve daha sürdürülebilir kod sayesinde)
- Yazılım geliştirme maliyet ve süresinin düşürülmesi

- Testlerin maliyet ve süresinin düşürülmesi
- Yaşam döngüsünün ilerleyen kısımlarında veya canlıya alımdan sonra daha az arıza olması sebebiyle yazılımın kullanım ömrü boyunca gerçekleşecek olan toplam kalite maliyetinin düşürülmesi
- Gözden geçirmeler sırasında ekip üyeleri arasındaki iletişimin iyileştirilmesi

3.1.3 Statik ve Dinamik Testler Arasındaki Farklar

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

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

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

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

- Gereksinim hataları (örneğin tutarsızlıklar, belirsizlikler, çelişkiler, çıkarmalar, yanlışlıklar ve fazlalıklar)
- Tasarım hataları (örneğin verimsiz algoritmalar veya veritabanı yapıları, yüksek bağlaşım, düşük uyum)
- Kodlama hataları (örneğin, değer atanmamış değişkenler, tanımlanmış fakat hiç kullanılmamış değişkenler, ulaşılamayan kod, tekrarlanan kod)
- Standartlardan sapmalar (örneğin, kodlama standartlarına bağlı kalmama)
- Hatalı arayüz gereksinimleri (örneğin, çağıran sistem ile çağrılan sistem tarafından kullanılan ölçüm birimlerinin farklı olması)
- Güvenlik açıkları (örneğin, arabellek aşımlarına karşı duyarlılık)
- Test esası izlenebilirliğinde veya kapsamında boşluklar veya yanlışlıklar (örneğin bir kabul kriteri için testlerin eksik olması)

maintainability defects

- Ayrıca, çoğu sürdürülebilirlik hatası sadece statik testler (örneğin, hatalı modülerleştirme, birimlerin tekrar kullanılabilirliğinin düşük olması, analiz edilmesi ve değiştirilmesi zor olan kodlar) ile bulunabilir.

3.2 Gözden Geçirme Süreci Review Process

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

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

ISO standardı (ISO/IEC 20246), çalışma ürünleri için roller ve gözden geçirme teknikleri de dâhil olmak üzere gözden geçirme sürecinin daha ayrıntılı açıklamalarını içerir.

3.2.1 Çalışma Ürünü Gözden Geçirme Süreci

Gözden geçirme süreci aşağıdaki ana aktivitelerden oluşur:

Planlama Planning

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

Gözden geçirmenin başlatılması Initiate review

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

Bireysel gözden geçirme (bireysel hazırlanma) Individual review (i.e., individual preparation)

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

Bulguların iletilmesi ve analizi Issue communication and analysis

- Belirlenen olası hataların iletilmesi (örneğin bir gözden geçirme toplantısında)
- Potansiyel hataların analiz edilmesi, bunlar için bir sorumlu ve durumun atanması
- Kalite özelliklerinin değerlendirilmesi ve dokümante edilmesi
- Gözden geçirme kararı (reddedildi; büyük değişiklikler gerekli; olası küçük değişikliklerle kabul edildi) vermek için gözden geçirme bulgularının çıkış kriterlerine göre değerlendirilmesi

Hataların giderilmesi ve raporlama Fixing and reporting

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

Bir çalışma ürünü gözden geçirmesinin sonuçları, bölüm 3.2.3'te de açıklandığı üzere gözden geçirme türüne ve resmi olup olmamasına bağlı olarak değişir.

3.2.2 Resmi Gözden Geçirmede Roller ve Sorumluluklar Roles and responsibilities

Tipik bir resmi gözden geçirme aşağıdaki rolleri içerir:

Yazar (çalışma ürünü yazarı/oluşturucu) Author

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

Yönetim Management

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

Moderatör Facilitator

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

Gözden geçirme lideri Review leader

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

Gözden geçiriciler Reviewers

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

Yazıcı (veya kaydedici) Scribe (or recorder)

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

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

Ayrıca, ISO standardında (ISO/ EC 20246) tanımlandığı gibi daha ayrıntılı roller de mümkündür.

3.2.3 Gözden Geçirme Çeşitleri Review Types

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

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

Informal review (e.g., buddy check, pairing, pair review)

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

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

Üzerinden geçme Walkthrough

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

Teknik gözden geçirme Technical review

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

Teftiş Inspection

- Ana amaçlar: Potansiyel hataların bulunması, çalışma ürünündeki kaliteyi değerlendirmek ve güven oluşturmak, çalışma ürününü yazanların öğrenmesi ve kök neden analizi yoluyla gelecekteki benzer hataları önlemek.

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

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

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

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

3.2.4 Gözden Geçirme Tekniklerinin Uygulanması

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

Kurgusuz — Ad hoc

Kurgusuz gözden geçirmede gözden geçiricilere bu görevin nasıl gerçekleştirilmesi gerektiği hakkında çok az rehberlik sağlanır veya hiç sağlanmaz. Gözden geçiriciler genellikle çalışma ürününü sırayla okur ve karşılaştıkları sorunları dokümante eder. Kurgusuz gözden geçirme çok az hazırlık gerektiren ve yaygın kullanılan bir tekniktir. Bu teknik büyük ölçüde gözden geçiricilerin becerilerine bağlıdır ve farklı gözden geçiriciler tarafından bildirilen birçok benzer sorunun bulunmasına yol açabilir.

Kontrol listesine dayalı — Checklist-based

Kontrol listesine dayalı gözden geçirme gözden geçiricilerin gözden geçirme başlangıcında (örneğin moderatör tarafından) dağıtılan kontrol listelerine dayanarak sorunları belirlediği sistematik bir tekniktir. Gözden geçirme kontrol listesi deneyimle elde edilen ve olası hatalara dayanan bir dizi sorudan oluşur. Kontrol listeleri gözden geçirilmekte olan çalışma ürününe özel olmalı ve önceki gözden geçirmelerde kaçırılan sorun türlerini kapsayacak şekilde düzenli olarak güncellenmelidir. Kontrol listesine dayalı tekniğin en önemli avantajı tipik hata çeşitlerinin sistematik bir şekilde kapsanmasıdır. Bireysel gözden geçirmelerde sadece kontrol listesini takip etmekle yetinmemeye ve kontrol listesinin dışındaki hataları da aramaya özen gösterilmelidir.

Senaryolar ve provalar — Scenarios and dry runs

Senaryoya dayalı gözden geçirmede gözden geçiricilere çalışma ürününü nasıl okuyacaklarına ilişkin rehberler sağlanır. Senaryoya dayalı yaklaşım, (çalışma ürünü, kullanım senaryoları gibi uygun bir biçimde dokümante edilmişse) çalışma ürünü üzerinde "provalar" yapılmasını sağlayarak gözden geçiricilerin işini kolaylaştırır. Bu senaryolar, gözden geçiricilere hata çeşitlerini bulma konusunda basit kontrol listelerinden daha iyi rehberlik sağlar.
Kontrol listesine dayalı gözden geçirmelerde olduğu gibi diğer hata çeşitlerini (örneğin, eksik özellikler) kaçırmamak için,

gözden geçiricilerin dokümante senaryolarla sınırlandırılmaması gerekir.

Role dayalı

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

Bakış açısına dayalı Perspective-based

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

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

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

3.2.5 Gözden Geçirmelerin Başarı Faktörleri Success Factors for Reviews

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

Gözden geçirmenin organizasyonel başarı faktörleri aşağıdaki gibidir:

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

Gözden geçirmelerin kişilere bağlı başarı faktörleri aşağıdaki gibidir:

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

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

Başarılı gözden geçirmeler hakkında daha fazla bilgi için Gilb 1993, Wiegers 2002 ve van Veenendaal 2004'e bakınız.

4. Test Tasarım Teknikleri — 330 dakika

Anahtar kelimeler

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

Test Teknikleri için Öğrenme Hedefleri

4.1 Test Tekniklerinin Kategorileri

black-box test techniques, white-box test techniques, and experience-based test techniques

FL-4.1.1 (K2) Kara kutu test teknikleri, beyaz kutu test teknikleri ve tecrübeye dayalı test tekniklerinin özelliklerini, ortak yanlarını ve aralarındaki farkları açıklayın

4.2 Kara Kutu Test Teknikleri

FL-4.2.1 (K3) Test senaryolarını verilen gereksinimlerden üretmek için denklik paylarına ayırın — 1) equivalence partitioning

FL-4.2.2 (K3) Test senaryolarını verilen gereksinimlerden üretmek için sınır değer analizi uygulayın — 2) boundary value analysis

FL-4.2.3 (K3) Test senaryolarını verilen gereksinimlerden üretmek için karar tablosu testi uygulayın — 3) decision table testing

FL-4.2.4 (K3) Test senaryolarını verilen gereksinimlerden üretmek için durum geçişi testi uygulayın — 4) state transition testing

FL-4.2.5 (K2) Test senaryosunun kullanım senaryosundan nasıl elde edileceğini açıklayın — 5) use case

4.3 Beyaz Kutu Test Teknikleri

FL-4.3.1 (K2) Komut kapsamını açıklayın — 1) statement coverage

FL-4.3.2 (K2) Karar kapsamını açıklayın — 2) decision coverage

FL-4.3.3 (K2) Komut ve karar kapsamının önemini açıklayın

4.4 Tecrübeye Dayalı Test Teknikleri — Experience-based test techniques

FL-4.4.1 (K2) Hata tahminlemeyi açıklayın — 1) error guessing

FL-4.4.2 (K2) Keşif testlerini açıklayın — 2) exploratory testing

FL-4.4.3 (K2) Kontrol listesine dayalı testi açıklayın — 3) checklist-based testing

4.1 Test Tekniklerinin Kategorileri

Test tekniğinin amacı, test koşullarını, test senaryolarını ve test verilerini belirlemeye yardımcı olmaktır.

4.1.1 Test Tekniklerinin Seçimi

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

- Birim veya sistem tipi
- Birim veya sistem karmaşıklığı
- Yasal standartlar Regulatory standards
- Müşteri veya sözleşme gereksinimleri
- Risk seviyeleri
- Risk çeşitleri
- Test hedefleri
- Mevcut dokümantasyon
- Test uzmanının bilgi ve yetenekleri Tester knowledge and skills
- Kullanılabilir araçlar Available tools
- Süre ve bütçe
- Yazılım geliştirme yaşam döngüsü modeli
- Yazılımın beklenen kullanımı
- Test edilecek birim veya sistem üzerinde test tekniklerini kullanma konusunda geçmiş deneyim
- Birim veya sistemde beklenen hata çeşitleri

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

Test analizi, test tasarımı ve test uyarlama aktivitelerinde test tekniklerinin kullanımı, gayri resmi (çok az veya sıfır dokümantasyon) ile resmi arasında değişebilir. Uygun resmilik seviyesi; test ve yazılım geliştirme süreçlerinin olgunluğu, zaman kısıtlamaları, emniyet veya yasal gereksinimler, proje ekibinin bilgi ve becerileri ve takip edilen yazılım geliştirme yaşam döngüsü modeli de dâhil olmak üzere test projesinin ve projenin bağlamına bağlıdır.

4.1.2 Test Tekniklerinin Kategorileri ve Karakteristik Özellikleri

Bu ders programında test teknikleri kara kutu, beyaz kutu veya tecrübeye dayalı olarak sınıflandırılmaktadır.

behavior-based techniques

Kara kutu test teknikleri (davranışsal veya davranışa dayalı teknikler olarak da bilinir) uygun bir test esasının (örneğin gereksinim dokümanları, spesifikasyonlar, kullanım senaryoları, kullanıcı hikâyeleri veya iş süreçleri) analizine dayanır. Bu teknikler hem fonksiyonel hem de fonksiyonel olmayan testlere uygulanabilir. Kara kutu test teknikleri, test nesnesinin iç yapısını dikkate almadan test nesnesinin girdi ve çıktılarına odaklanır.

structure-based techniques

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

experience-based techniques

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

Kara kutu test tekniklerinin ortak özellikleri aşağıda verilmiştir:

- Test koşulları, test senaryoları ve test verileri; yazılım gereksinimlerini, spesifikasyonları, kullanım senaryolarını ve kullanıcı hikâyelerini içeren bir test esasından elde edilir.

- Gereksinimlerden sapmaların yanı sıra gereksinimler ve gereksinimlerin hayata geçirilmesi arasındaki boşlukları belirlemek için test senaryoları kullanılabilir.
- Kapsam, test esasındaki test edilen öğelere ve test esasına uygulanan tekniğe göre ölçülür.

Beyaz kutu test tekniklerinin ortak özellikleri aşağıda verilmiştir:

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

Tecrübeye dayalı test tekniklerinin ortak özellikleri aşağıda verilmiştir:

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

Bu bilgi ve tecrübeye, yazılımın beklenen kullanımı, ortamı, muhtemel hataları ve bu hataların dağılımı dâhildir. Uluslararası standart (ISO/IEC/IEEE 29119-4), test tekniklerinin açıklamalarını ve bunlara karşılık gelen kapsam ölçülerini içerir (teknikler hakkında daha fazla bilgi için bkz. Craig 2002 ve Copeland 2004).

4.2 Kara Kutu Test Teknikleri

4.2.1 Denklik Paylarına Ayırma Equivalence Partitioning

Denklik paylarına ayırma, verileri paylara (denklik sınıfları) ayırır; yazılım tarafından bir payın tüm üyelerinin benzer şekilde ele alınması beklenir (bkz. Kaner 2013 ve Jorgensen 2014). Hem geçerli hem de geçersiz değerler için denklik payları vardır.

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

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

4.2.2 Sınır Değer Analizi Boundary Value Analysis

Sınır değer analizi, denklik paylarına ayırma tekniğinin genişletilmiş halidir, ancak yalnızca sayısal veya sıralı verilerden oluşan paylarda kullanılabilir. Bir payın minimum ve maksimum değerleri (veya ilk ve son değerleri) sınır değerleridir (Beizer 1990).

Örneğin, bir giriş alanının tek bir tam sayı değerini girdi olarak kabul ettiğini varsayalım, tam sayı olmayan girdilerin imkânsız olması için girdileri sınırlamakta bir tuş takımı kullanılıyor olsun. Geçerli aralık 1 ile 5 arasındadır, sınır değerleri de dâhildir.

Dolayısıyla, üç denklik payı vardır: geçersiz (alt); geçerli; geçersiz (üst). Geçerli denklik payı için sınır değerleri 1 ve 5'tir. Geçersiz (üst) denklik payı için sınır değerleri 6 [redacted]dır. Geçersiz (alt) pay için, yalnızca bir sınır değeri vardır, o da 0'dır, çünkü bu yalnızca bir üyesi olan bir paydır.

Yukarıdaki örnekte, sınır başına iki sınır değeri tanımlarız. Geçersiz (alt) ve geçerli arasındaki sınır, test değerleri olarak 0 ve 1'i verir. Geçerli ve geçersiz (üst) arasındaki sınır, test değerleri olarak 5 ve 6'yı verir. Bu tekniğin bazı varyasyonları, sınır başına üç sınır değeri tanımlar: sınırdan önceki, sınırdaki ve hemen sınırdan sonraki değerler. Önceki örnekte, üç noktalı sınır değerleri kullanıldığında, alt sınır testi değerleri 0, 1 ve 2'dir ve üst sınır testi değerleri 4, 5 ve 6'dır (Jorgensen 2014).

Denklik paylarının sınırlarındaki davranışların hatalı olma olasılığı, payların içinden seçilip test edilecek verilere göre daha yüksektir. Yazılımın riskine göre sınır değerler üzerinde oynamalar yapmak; bunları kaydırmak, sınır başına seçilen sınır değer sayısını artırmak veya azaltmak mümkündür.
Sınır değer analizi ve testleri, test verilerini sınır değerin ait olduğu payın dışında farklı bir paydan seçerek yazılımın davranışlarını zorlamaktadır.

Sınır değer analizi tüm test seviyelerinde uygulanabilir. Bu teknik genellikle bir sayı aralığı (tarihler ve saatler dâhil) gerektiren gereksinimleri test etmek için kullanılır. Kapsam, test edilen sınır değerlerin sayısının tüm sınır değerlerin sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

4.2.3 Karar Tablosu Testleri Decision Table Testing

Karar tablosu testlerinin de yer aldığı kombinasyonlu test teknikleri farklı test koşulları kombinasyonlarının nasıl farklı sonuçlar verdiğini test etmek için kullanılır.

Karar tabloları, bir yazılımın ele alması beklenen karmaşık iş kurallarını kaydetmenin iyi bir yoludur. Karar tabloları oluştururken, test uzmanı yazılımın koşullarını (genellikle girdiler) ve sonuçta ortaya çıkan aksiyonları (genellikle çıktılar) tanımlar. Bunlar tablonun satırlarını oluşturur, genellikle koşullar üstte ve aksiyonlar alttadır. Her sütun bir karar kuralına karşılık gelir; bu kural, kendisiyle ilişkili aksiyonların gerçekleştirilmesini sağlayan özgün bir koşul kombinasyonunu içerir. Koşulların ve aksiyonların değerleri genellikle mantıksal değerler (boolean - doğru veya yanlış) veya ayrık değerler (örneğin kırmızı, yeşil, mavi) olarak gösterilir, ancak sayılar veya sayı aralıkları da olabilir. Bu farklı türlerdeki koşullar ve aksiyonlar aynı tabloda bir arada bulunabilir.

Karar tablolarındaki yaygın gösterim aşağıdaki gibidir:

Koşullar için:

- Y, koşulun doğru olduğu anlamına gelir (ayrıca T veya 1 olarak da gösterilebilir)
- N, koşulun yanlış olduğu anlamına gelir (ayrıca F veya 0 olarak da gösterilebilir)
- — koşulun değerinin önemli olmadığı anlamına gelir (N/A olarak da gösterilebilir)

Aksiyonlar için:

- X, eylemin gerçekleşmesi gerektiği anlamına gelir (ayrıca Y veya T veya 1 olarak da gösterilebilir)
- Boşluk, eylemin gerçekleşmemesi gerektiği anlamına gelir (ayrıca veya N veya F veya 0 olarak da gösterilebilir)

Tam bir karar tablosunda her koşul kombinasyonunu kapsayacak kadar sütun vardır. İmkânsız koşul kombinasyonlarını içeren sütunları, olası ancak elverişsiz koşul kombinasyonlarını içeren sütunları ve çıktıyı etkilemeyen koşulların kombinasyonlarını test eden sütunları silinerek tablo küçültülebilir. Karar tablolarının nasıl küçültüleceği hakkında daha fazla bilgi için, bkz. ISTQB-ATA İleri Seviye Test Analisti Ders Programı.

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

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

4.2.4 Durum Geçişi Testleri State Transition Testing

Birimler veya sistemler, mevcut veya geçmiş durumuna (örneğin, sistemin başlatılmasından bu yana gerçekleşen olaylar) bağlı olarak gerçekleşen bir olaya farklı şekilde yanıt verebilir. Bir sistemin geçmiş davranışı, durumlar kavramı kullanılarak özetlenebilir. Durum geçişi diyagramı, olası yazılım durumlarının yanı sıra, yazılımın ilgili durumlara nasıl girdiğini, çıktığını

ve aralarındaki geçişleri gösterir. Geçiş, olay tarafından başlatılır (örneğin, bir değerin bir alana kullanıcı tarafından girilmesi).

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

Durum geçişi tablosu, durumlar arasındaki tüm geçerli geçişleri ve potansiyel geçersiz geçişleri, ayrıca olayları, koruma koşullarını ve geçerli geçişler için aksiyonları gösterir. Durum geçiş diyagramları normalde yalnızca geçerli geçişleri gösterir ve geçersiz geçişleri ele almaz.

Tipik bir durum dizisini kapsamak, tüm durumları denemek, her geçişi denemek, belirli geçiş dizilerini denemek veya geçersiz geçişleri test etmek için testler tasarlanabilir.

Durum geçişi testleri, menü bazlı uygulamalar için kullanılabilir ve gömülü yazılım endüstrisinde yaygın bir şekilde kullanılmaktadır. Bu teknik ayrıca belirli durumlara sahip bir iş senaryosunu modellemek veya ekran geçişlerini test etmek için de uygundur. Durum, soyut bir kavramdır - birkaç kod satırını veya bütün bir iş sürecini temsil edebilir.

Kapsam, genel olarak test edilen durum veya geçişlerin sayısının, test nesnesindeki durumların veya geçişlerin toplam sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir. Durum geçişi testlerinin kapsam kriterleri hakkında daha fazla bilgi için, bkz. ISTQB-ATA İleri Seviye Test Analisti Ders Programı.

4.2.5 Kullanım Senaryosu Testleri Use Case Testing

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

Her kullanım senaryosu, bir sistemin bir veya daha fazla aktör ile işbirliği içinde gerçekleştirebileceği bazı davranışları belirtir (UML 2.5.1 2017). Bir kullanım senaryosu, etkileşimler ve aktivitelerin yanı sıra uygun durumlarda önkoşullar, artkoşullar ve doğal bir anlatımla tanımlanabilir. Aktörler ve sistem arasındaki etkileşimler sistemin durumunda değişikliklere neden olabilir. Etkileşimler; iş akışları, faaliyet şemaları veya iş süreç modelleri ile grafiksel olarak gösterilebilir.

Bir kullanım senaryosu, istisnai davranış ve hata ele alma dâhil olmak üzere, temel yazılım davranışının olası varyasyonlarını, alternatiflerini içerebilir. Testler, tanımlanmış davranışları (temel, alternatif, istisnai ve hata ele alma) denemek için tasarlanır. Kapsam, test edilen kullanım senaryosu davranışlarının toplam kullanım senaryosu davranışı sayısına bölünmesiyle ölçülür ve normalde yüzde olarak ifade edilir.

Kullanım senaryosu testlerinin kapsam kriterleri hakkında daha fazla bilgi için, bkz. ISTQB-ATA İleri Seviye Test Analisti Ders Programı.

4.3 Beyaz Kutu Test Teknikleri

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

4.3.1 Komut Testi ve Kapsamı Statement Testing and Coverage

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

4.3.2 Karar Testi ve Kapsamı Decision Testing and Coverage

Karar testi, koddaki kararların üzerinden geçilip bu kararların çalıştırılması ve karar çıktılarına dayanarak kodun test edilmesidir. Bunun için test senaryoları karar noktasındaki kontrol akışlarını takip eder (örneğin, bir IF komutu için, biri doğru çıktı ve biri yanlış çıktı; bir CASE komutu için tüm olası çıktıların test senaryoları gerekli olacaktır).

Kapsam, testler tarafından çalıştırılan karar çıktılarının sayısının test nesnesindeki karar çıktılarının toplam sayısına bölünmesi ile ölçülür ve normalde yüzde olarak ifade edilir.

4.3.3 Komut ve Karar Testlerinin Önemi — The value of **statement** and **decision** testing

%100 komut kapsama yüzdesi sağlandığında koddaki tüm çalıştırılabilir komutların en az bir kez test edilmesi sağlanır, ancak tüm kararların tamamen test edilmesi sağlanmaz. Bu ders programında bahsedilen iki beyaz kutu tekniği arasında komut testleri, karar testlerine kıyasla daha az kapsama sağlayabilir.

%100 karar kapsamı sağlandığında, tüm karar çıktıları çalıştırılır; buna, doğru çıktının ve yanlış çıktının test edilmesi de dâhildir. Komut kapsama yüzdesi, kodda, diğer testler tarafından denenmemiş hataların bulunmasına yardımcı olur. Karar kapsamı, ise diğer testlerin ele almadığı hem doğru hem de yanlış çıktıların ele alınmasını sağlar.

%100 karar kapsamına ulaşılması %100 komut kapsamına ulaşılmasını garanti eder (ancak bunun tersi geçerli değildir).

4.4 Tecrübeye Dayalı Test Teknikleri — Experience-based Test Techniques

Tecrübeye dayalı test teknikleri uygulanırken, test senaryoları, test uzmanının beceri ve sezgilerinden ve benzer uygulama ve teknolojilerdeki tecrübelerinden elde edilir. Bu teknikler, diğer daha sistematik tekniklerle kolayca bulunamayan hataları bulmada yardımcı olabilir. Test uzmanının yaklaşımına ve tecrübesine bağlı olarak, bu teknikler çok çeşitli kapsam ve etkinlik derecelerine ulaşabilir. Kapsamın değerlendirilmesi zor olabilir ve bu tekniklerle ölçülemeyebilir.

Yaygın olarak kullanılan tecrübeye dayalı teknikler aşağıdaki bölümlerde ele alınmaktadır.

4.4.1 Hata Tahminleme — Error Guessing

Hata tahminleme, test uzmanının bilgilerine dayalı olarak yanlışların, hataların ve arızaların oluşmasını sağlamak için kullanılan bir tekniktir; bu bilgiler aşağıdaki gibidir:

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

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

4.4.2 Keşif Testi — Exploratory Testing

Keşif testlerinde, test koşumu sırasında gayri resmi (önceden tanımlanmamış) testler tasarlanır, koşturulur, kaydedilir ve dinamik olarak değerlendirilir. Test sonuçları, birim veya sistem hakkında daha fazla bilgi edinmek ve daha fazla test gerektirebilecek alanlar için testler oluşturmak için kullanılır.

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

Keşif testleri, gereksinimler az veya yetersiz olduğunda veya testler üzerinde önemli bir zaman baskısı olduğunda çok işe yarar. Keşif testleri, diğer daha resmi test tekniklerini tamamlamak için de kullanılır.

Keşif testleri, tepkisel test stratejileri ile güçlü bir şekilde ilişkilidir (bkz. Bölüm 5.2.2). Keşif testleri, diğer kara kutu, beyaz kutu ve tecrübeye dayalı tekniklerle birlikte kullanılabilir.

4.4.3 Kontrol Listesine Dayalı Testler — Checklist-based Testing

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

Fonksiyonel ve fonksiyonel olmayan testleri desteklemek için kontrol listeleri oluşturulabilir. Ayrıntılı test senaryoları olmadığında, kontrol listesine dayalı testler rehberlik ve belirli bir yere kadar tutarlılık sağlayabilir. Bunlar üst seviye listeler

olduğu için, gerçek testlerde bazı değişikliklerin ortaya çıkması muhtemeldir; bu da potansiyel olarak daha büyük kapsama ama daha düşük tekrarlanabilirliğe yol açar.

5. Test Yönetimi (K3) — 170 dakika

Anahtar kelimeler

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

Test Yönetimi için Öğrenme Hedefleri

5.1 Test Organizasyonu

FL-5.1.1 (K2) Bağımsız testlerin yararlarını ve sakıncalarını açıklayın

FL-5.1.2 (K1) Test yöneticisi ve test uzmanının görevlerini tanımlayın

5.2 Test Planlama ve Tahminleme

FL-5.2.1 (K2) Bir test planının amacını ve içeriğini özetleyin

FL-5.2.2 (K2) Çeşitli test stratejilerinin birbirinden farklarını belirtin

FL-5.2.3 (K2) Potansiyel giriş ve çıkış kriterlerine örnekler verin

FL-5.2.4 (K3) Belirli bir test senaryosu grubunun test koşumunu planlamak için önceliklendirme kriterlerini ve teknik ve mantıksal bağımlılıkları uygulayın.

FL-5.2.5 (K1) Testlerle ilgili çalışmaları etkileyen faktörleri tanımlayın

FL-5.2.6 (K2) İki tahminleme tekniği arasındaki farkı açıklayın: metrik bazlı teknik ve uzman bazlı teknik

5.3 Test Gözetimi ve Kontrolü

FL-5.3.1 (K1) Testlerde kullanılan metrikleri anımsayın

FL-5.3.2 (K2) Test raporlarının amaçlarını, içeriklerini ve hedef kitlelerini özetleyin

5.4 Yapılandırma Yönetimi

FL-5.4.1 (K2) Yapılandırma yönetiminin testleri nasıl desteklediğini özetleyin

5.5 Riskler ve Testler

FL-5.5.1 (K1) Olasılık ve etkiyi kullanarak risk seviyesini tanımlayın

FL-5.5.2 (K2) Proje ve ürün risklerinin farklarını belirtin

FL-5.5.3 (K2) Ürün risk analizinin testlerin bütünlüğünü ve kapsamını nasıl etkileyebileceğini örnekler vererek açıklayın

5.6 Hata Yönetimi

FL-5.6.1 (K3) Testler sırasında belirlenen hataları kapsayan bir hata raporu yazın

5.1 Test Organizasyonu

5.1.1 Testte Bağımsızlık Independent Testing

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

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

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

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

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

Test bağımsızlığının potansiyel faydaları aşağıda sıralanmıştır:

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

Test bağımsızlığının potansiyel sakıncaları aşağıda sıralanmıştır:

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

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

5.1.2 Test Yöneticisi ve Test Uzmanının Görevleri Tasks of a Test Manager and Tester

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

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

Tipik test yöneticisinin görevleri aşağıdakileri içerebilir:

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

Test yöneticisi rolünün yerine getirilme şekli, yazılım geliştirme yaşam döngüsü modeline göre değişir. Örneğin, çevik geliştirmede, yukarıda belirtilen görevlerin bazıları çevik ekip tarafından gerçekleştirilir; özellikle ekip içinde yapılan günlük testlerle ilgili görevler, genellikle ekip içinde çalışan bir test uzmanı tarafından yerine getirilir. Birden fazla ekibi veya tüm kurumu kapsayan veya personel yönetimi ile ilgili olan bazı görevler, yazılım geliştirme ekibi dışından test yöneticileri tarafından yapılabilir; bunlara bazen test koçları denir.
Test sürecinin yönetimi hakkında daha fazla bilgi için Black 2009'a bakınız.

Test uzmanının tipik görevleri aşağıdakileri içerebilir:

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

- Performans, güvenilirlik, kullanılabilirlik, güvenlik, uyumluluk ve taşınabilirlik gibi fonksiyonel olmayan testleri yapmak
- Diğerleri tarafından yapılan testleri gözden geçirmek

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

5.2 Test Planlama ve Tahminleme Test Planning and Estimation

5.2.1 Test Planının Amacı ve İçeriği Purpose and Content of a Test Plan

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

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

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

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

5.2.2 Test Stratejisi ve Test Yaklaşımı Test Strategy and Test Approach

Test stratejisi, test sürecinin genellikle ürün veya kurum düzeyinde genel bir tanımını sağlar. Yaygın test stratejisi çeşitleri aşağıda verilmiştir:

- Analitik: Bu tür test stratejileri, bazı faktörlerin (örneğin, gereksinim veya risk) analizine dayanır. Risk bazlı testler, testlerin risk seviyesine göre tasarlandığı ve önceliklendirildiği analitik bir yaklaşım örneğidir. Analytical
- Model Bazlı: Bu tür test stratejilerinde testler; fonksiyon, iş süreci, iç çalışma yapısı veya fonksiyonel olmayan bir özellik (örneğin güvenilirlik) gibi, ürünün bazı özelliklerinin modellenmesine dayanarak tasarlanır. Bu modellerin Model-Based

örnekleri arasında iş süreci modelleri, durum modelleri ve güvenilirlik büyüme modelleri bulunur.

- Methodical: Bu tür test stratejileri; yaygın veya muhtemel arıza türlerinin sınıflandırması, önemli kalite özelliklerinin listesi veya kurum genelinde mobil uygulamalar veya web sayfaları için görünüm ve çalışma standartları gibi bazı önceden tanımlanmış test kümelerinin veya test koşullarının sistematik olarak kullanılmasına dayanır.

- Process-compliant (or standard-compliant): Süreç uyumluluk (veya standartlara uyumluluk): Bu tür test stratejileri; dış yönergeleri ve standartları baz alarak testlerin analiz edilmesini, tasarlanmasını ve uyarlanmasını içerir. Bu kurallar ve standartlara örnek olarak sektörel standartlar, süreç dokümantasyonu, test esasının titiz bir şekilde tanımlanması ve kullanılması ile belirlenen veya kurumun uygulanmasını zorunlu tuttuğu veya kurumun uygulamasının zorunlu olduğu herhangi bir süreç veya standart olabilir.

- Directed (or consultative): Yönlendirmeli (veya danışılarak): Bu tür test stratejileri, temel olarak test ekibinin dışında veya kurum dışından paydaşların, alan uzmanlarının veya teknoloji uzmanlarının tavsiyesi, rehberliği veya talimatları ile yönlendirilir.

- Regression-averse: Regresyon hassasiyetli: Bu tür test stratejileri, yazılımın mevcut çalışan özelliklerinin bozulmasını (regresyon) engelleme amacıyla hayata geçirilmektedir. Bu test stratejisi, mevcut test yazılımının yeniden kullanımını (özellikle test senaryoları ve test verileri), geniş regresyon testi otomasyonunu ve standart test gruplarını içerir.

- Reactive: Tepkisel: Bu tür test stratejilerinde, testler, (önceki stratejiler gibi) önceden planlanmak yerine, test edilen birim veya sisteme ve test koşumu sırasında meydana gelen olaylara verilen tepkilerle şekillenir. Testler tasarlanır, uyarlanır ve önceki test sonuçlarından elde edilen deneyimler baz alınarak koşturulur. Keşif testleri, tepkisel stratejilerde kullanılan yaygın bir tekniktir.

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

Test stratejisi, test sürecinin genel bir tanımını sağlarken, test yaklaşımı belirli bir proje veya sürüm için test stratejisini düzenler. Test yaklaşımı; test tekniklerini, test seviyelerini ve test çeşitlerini seçmek ve giriş kriterlerini ve çıkış kriterlerini (veya sırasıyla Hazır tanımını ve Tamamlandı tanımını) tanımlamak için başlangıç noktasıdır. Stratejinin uyarlanması; projenin karmaşıklığı ve hedefleri, geliştirilen ürünün türü ve ürün riski analizi ile ilgili olarak verilen kararlara dayanır. Seçilen yaklaşım proje ve test bağlamına bağlıdır ve riskler, emniyet, mevcut kaynaklar ve beceriler, teknoloji, sistemin yapısı (örneğin, ticari paket yazılım), test hedefleri ve düzenlemeler gibi faktörleri dikkate alabilir.

5.2.3 Giriş Kriterleri ve Çıkış Kriterleri (Hazır Tanımı ve Tamamlandı Tanımı)

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

Tipik giriş kriterleri aşağıda verilmiştir:

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

Tipik çıkış kriterleri aşağıda verilmiştir:

- Planlanan testlerin koşturulmuş olması
- Belirlenmiş bir kapsama seviyesine (örneğin, gereksinimler, kullanıcı hikâyeleri, kabul kriterleri, riskler, kod) ulaşılmış olması

- Çözülemeyen hataların sayısının önceden kararlaştırılmış bir limitin dahilinde olması
- Tahmini kalan hata sayısının yeterince düşük olması
- Değerlendirilen güvenilirlik seviyeleri, performans, kullanılabilirlik, güvenlik ve diğer ilgili kalite özelliklerinin yeterli olması

the budget being expended, the scheduled time being completed, and/or pressure to bring the product to market

Çıkış kriterleri karşılanmasa bile, bütçenin tükenmesi, planlanan süresinin tamamlanması ve/veya ürünün piyasaya sürülmesi için oluşan baskı nedeniyle test faaliyetlerinin kısa kesilmesi de yaygın bir durumdur. Proje paydaşları ve ürün sahipleri daha fazla test yapmadan ürünü piyasaya sürmenin riskini gözden geçirip kabul ettiyse, bu şartlar altında testlere son verilmesi kabul edilebilir bir durumdur.

5.2.4 Test Koşum Çizelgesi

Test Execution Schedule

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

İdeal olarak, test senaryoları öncelik seviyelerine göre çalışma sırasına alınır, genellikle en yüksek önceliğe sahip test senaryoları önce koşturulur. Ancak, test senaryolarının veya test edilen özelliklerin bağımlılıkları varsa, bu yöntem çalışmayabilir. Yüksek öncelikli bir test senaryosu düşük öncelikli bir test senaryosuna bağımlıysa, önce düşük öncelikli test senaryosu koşturulmalıdır.
Benzer şekilde, test senaryoları arasında bağımlılıklar varsa, göreceli öncelikleri ne olursa olsun uygun şekilde sıralama yapılmalıdır. Değişikliklerle ilgili hızlı geri bildirimin önemine dayanarak, onaylama ve regresyon testlerine de öncelik verilmelidir, ancak yine burada da bağımlılıklar geçerli olabilir.

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

5.2.5 Test Eforunu Etkileyen Faktörler

Factors Influencing the Test Effort

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

Ürünün özellikleri

Product characteristics

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

Yazılım geliştirme süreci özellikleri

Development process characteristics

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

Testi Yapacak Kişilerin özellikleri People characteristics

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

Test sonuçları Test results

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

5.2.6 Test Tahminleme Teknikleri Test Estimation Techniques

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

- Metrik bazlı teknik: önceki benzer projelerin metriklerine veya tipik değerlere dayanarak test eforunu tahmin etme
- Uzman bazlı teknik: testi gerçekleştirecek kişilerin tecrübesine veya uzmanlara dayanarak test eforunu tahmin etme

Örneğin, çevik yazılım geliştirmede, burndown tabloları (yapılacak işler tablosu), metrik bazlı yaklaşımın örneklerindendir; çalışma eforu bu tablolara kaydedilip raporlanır ve daha sonra ekibin bir sonraki döngüde yapabileceği iş miktarını belirlemek için ekibin hızını düzenlemekte kullanılır; poker planlama (Wideband Delphi tahminleme tekniğini baz alır) ise, ekip üyelerinin kendi deneyimlerine dayanarak yapılacak iş için gereken çalışmayı tahmin ettikleri bir tekniktir (ISTQB-AT Temel Seviye Çevik Test Uzmanı Devam Kursu Ders Programı).

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

5.3 Test Gözetimi ve Kontrolü Test Monitoring and Control

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

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

Test kontrol eylemlerine örnekler aşağıda sıralanmıştır:

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

5.3.1 Yazılım Testlerinde Kullanılan Metrikler Metrics Used in Testing

Aşağıdakileri değerlendirmek için test faaliyetleri sırasında ve sonunda metrikler toplanabilir:

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

- Test faaliyetlerinin hedeflere göre etkinliği

Yaygın test metrikleri aşağıdakileri içerir:

- Test senaryosu hazırlamada yapılan planlı işlerin yüzdesi (veya koşulan planlı test senaryolarının yüzdesi)
- Test ortamı hazırlamada yapılan planlı işlerin yüzdesi
- Test senaryosunun koşturulması (örneğin, çalıştırılmış/çalıştırılmamış test senaryolarının sayısı, başarılı/başarısız test senaryoları ve/veya başarılı/başarısız test koşulları)
- Hata bilgileri (örneğin hata yoğunluğu, belirlenen ve çözülen hatalar, arıza oranı ve onaylama testi sonuçları)
- Gereksinimlerin, kullanıcı hikâyelerinin, kabul kriterlerinin, risklerin veya kodun test kapsamı
- Görev tamamlama, kaynak tahsisi, kaynak kullanımı ve çalışma
- Maliyet/hata bulma faydasının karşılaştırılması veya maliyet/testi çalıştırma faydasının karşılaştırılması da dâhil olmak üzere testlerin maliyeti

Purposes, Contents, and Audiences for Test Reports

5.3.2 5.3.1 Test Raporlarının Amaçları, İçeriği ve Hedef Kitlesi

Test raporlamanın amacı, bir test faaliyeti sırasında ve sonunda (örneğin bir test seviyesi) test faaliyeti bilgilerini özetlemek ve iletmektir. Test faaliyeti sırasında hazırlanan test raporuna test ilerleme raporu, test faaliyeti sonunda hazırlanan test raporuna ise test özet raporu denebilir.

test progress report

test summary report

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

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

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

[redacted] test özet raporları genellikle aşağıdakileri içerir:

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

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

Test raporlarını projenin bağlamına göre uyarlanmanın yanı sıra rapor hedef kitlesine göre de düzenlenmelidir. Teknik bir hedef kitleye veya bir test ekibine yönelik verilen bilgilerin türü ve miktarı, yönetici özet raporunda verilenden farklı olabilir. İlk durumda, hata çeşitleri ve eğilimleri hakkında ayrıntılı bilgiler önemli olabilir. İkinci durumda, üst düzey bir rapor (ör. öncelik sırasına göre hataların, bütçenin, zaman çizelgesinin ve başarılı/başarısız/test edilmemiş test koşullarının durum özeti) daha uygun olabilir.

ISO standardı (ISO/IEC/IEEE 29119-3) test ilerleme raporları ve test tamamlama raporları (bu ders programında test özet

raporları olarak adlandırılmıştır) olmak üzere iki tür test raporundan bahseder ve iki rapor türü için de ana başlıklar ve örnekler içerir.

5.4 Yapılandırma Yönetimi Configuration Management

Yapılandırma yönetiminin amacı, birim veya sistemin, test yazılımının, proje ve ürün yaşam döngüsü boyunca birbirleriyle olan ilişkilerinin bütünlüğünü sağlamak ve korumaktır.

Testleri uygun şekilde desteklemek için yapılandırma yönetimi aşağıdakilerin gerçekleştirilmesini içerebilir:

- Tüm test öğeleri özgün bir şekilde tanımlanmış, versiyon kontrolü yapılmış, değişiklikleri izlenmiş ve birbirleriyle ilişkilendirilmiştir. All test items are uniquely **identified, version controlled**, **tracked for changes,** and related to each other
- Test için yazılan tüm yazılımlar benzersiz bir şekilde tanımlanmış, versiyon kontrolü yapılmış, birbirleriyle ve test öğesinin/öğelerinin versiyonlarıyla ilgili değişiklikler için izlenmiş, böylece test süreci boyunca izlenebilirlikleri korunmuştur.
- Tanımlanan tüm dokümanlar ve yazılım öğeleri, test dokümantasyonunda açık bir şekilde belirtilmiştir.

Test planlama sırasında yapılandırma yönetimi prosedürleri ve altyapısı (araçlar) tanımlanmalı ve uyarlanmalıdır.
During test planning, configuration management procedures and infrastructure (tools) should be identified and implemented.

5.5 Riskler ve Testler Risks and Testing

5.5.1 Risk Tanımı Definition of Risk

likelihood impact (the harm)

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

5.5.2 Ürün ve Proje Riskleri Product and Project Risks

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

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

Proje riski, gerçekleşmesi durumunda, projenin hedeflerine ulaşma imkânı üzerinde olumsuz etkiye sahip olabilecek durumları içerir. Proje risklerine örnekler aşağıda verilmiştir:

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

- Yetkinlik, eğitim ve personel yeterli olmayabilir
- Personel sorunları çatışma ve problemlere neden olabilir
- Çakışan işletme öncelikleri nedeniyle kullanıcılara, işletme personeline veya konunun uzmanlarına ulaşılamayabilir

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

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

5.5.3 Risk Bazlı Testler ve Ürün Kalitesi Risk-based Testing and Product Quality

Risk, test eforunu odaklamak için kullanılır. Testlere nerede ve ne zaman başlanacağını ve daha fazla dikkat gerektiren alanları belirlemek için kullanılır. Testler, olumsuz bir olay gerçekleşme olasılığını azaltmak veya olumsuz bir olayın etkilerini azaltmak için kullanılır. Testler, belirlenen risklerin riskini azaltma ve onlar hakkında geri bildirim sağlama, kalan (ele alınamamış) riskler hakkında ise geri bildirim sağlamak için kullanılır.

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

Risk bazlı bir yaklaşımda ürün risk analizinin sonuçları aşağıdaki amaçlar için kullanılır:

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

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

- Neyin yanlış gidebileceğinin (riskler) analizi (ve düzenli olarak yeniden değerlendirilmesi)
- Hangi risklerin önlenmesinin önemli olduğunun belirlenmesi

- Bu riskleri azaltmak için aksiyon alınması
- Gerçekleşmeleri halinde risklerle başa çıkmak için acil durum planlarının yapılması

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

5.6. Hata Yönetimi Defect Management

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

Hata yönetimi sürecinde bazen hatalardan kaynaklanan gerçek arızalar değil yanlış hatalar, yanlış pozitifleri raporlanabilir. Örneğin, bir ağ bağlantısı kesildiğinde veya süresi dolduğunda bir test başarısız olabilir. Bu başarısızlık, test nesnesindeki bir hatadan kaynaklanmamaktadır, ancak araştırılması gereken bir anomalidir. Test uzmanları, hata olarak bildirilen yanlış pozitiflerin sayısını azaltmaya çalışmalıdır.

Kodlama, statik analiz, gözden geçirmeler, dinamik testler veya yazılım ürününün kullanımı sırasında hatalar bulunabilir. Gereksinimler, kullanıcı hikâyeleri ve kabul kriterleri, yazılım geliştirme dokümanları, test dokümanları, kullanıcı kılavuzları veya kurulum kılavuzları dâhil olmak üzere herhangi bir dokümantasyon türü içindeki, koddaki veya sistemdeki sorunlar için hatalar bildirilebilir. Kurumlar, etkin ve verimli bir hata yönetimi sürecine sahip olmak amacıyla hataların nitelikleri, sınıflandırması ve iş akışı için standartlar tanımlayabilir.

Arıza raporlarının amaçları genellikle aşağıdaki gibidir:

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

Dinamik testler sırasında oluşturulan bir hata raporu genellikle aşağıdakileri içerir:

- Hatanın seri numarası veya sırası
- Bulunan hatanın başlığı ve kısa bir özeti
- Hata raporunun tarihi, düzenleyen kurum ve yazan kişi
- Test öğesinin (test edilen yapılandırma öğesi) ve ortamının tanımı
- Hatanın bulunduğu yazılım geliştirme yaşam döngüsü aşaması/aşamaları
- Günlükler, veri tabanı ekran görüntüleri veya kayıtlar (test koşumu sırasında bulunursa) dâhil olmak üzere hatanın yeniden oluşturulmasına ve çözülmesine olanak sağlamak için hatanın tanımı
- Beklenen ve gerçekleşen sonuçlar
- Hatanın kapsamı veya paydaşlar üzerindeki etki derecesi (önem derecesi)
- Düzeltilme aciliyeti/önceliği
- Hata raporunun durumu (ör. açık, ertelenmiş, tekrarlanmış, düzeltilmeyi bekliyor, onaylama testleri bekleniyor, yeniden açıldı, kapatıldı)
- Sonuçlar, öneriler ve onaylar
- Hatadan kaynaklanan bir değişiklikten etkilenebilecek diğer alanlar gibi genel sorunlar

- Değişiklik geçmişi: ör. proje ekibi üyeleri tarafından gerçekleştirilen, hatayı bulma, giderme ve giderildiğini doğrulama işlemleri dizisi
- Problemi ortaya çıkaran test senaryosu da dâhil olmak üzere verilen referanslar

Bu detaylardan bazıları, hata yönetimi araçlarını kullanırken otomatik olarak üretilebilir; ör. bir seri numarasının otomatik olarak hataya atanması, iş akışı sırasında hata raporu durumunun atanması ve güncellenmesi gibi. Statik testler, özellikle de gözden geçirmeler sırasında bulunan hatalar, normal olarak farklı bir şekilde dokümante edilir, ör. gözden geçirme toplantısı notlarında.

Örnek bir hata raporu içeriği, ISO standardında bulunabilir (ISO/IEC/IEEE 29119-3) (hata raporlarını olay raporları olarak adlandırır).

6. Yazılım Test Araç Desteği | 40 dakika

Anahtar kelimeler

data-driven testing, keyword-driven testing, test automation, test execution tool, t est management tool

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

Test Araçları için Öğrenme Hedefleri

6.1 Test araçlarındaki önemli hususlar

FL-6.1.1 (K2) Test araçlarını amaçlarına ve destekledikleri test faaliyetlerine göre sınıflandırın

FL-6.1.2 (K1) Test otomasyonunun yararlarını ve risklerini tanımlayın

FL-6.1.3 (K1) Test koşumu ve test yönetim araçlarına dair önemli hususları hatırlayın

6.2 Araçların etkin kullanımı

FL-6.2.1 (K1) Araç seçimi için temel prensipleri belirtin

FL-6.2.2 (K1) Araçları daha iyi anlamak için pilot proje kullanımının hedeflerini sayın

FL-6.2.3 (K1) Bir kurumdaki test araçlarının değerlendirilmesi, uyarlanması, dağıtımı ve desteğinin devam etmesi için başarı faktörlerini tanımlayın

6.1 Test Araçlarındaki Önemli Hususlar

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

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

6.1.1 Test Aracı Sınıflandırması

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

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

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

Bazı araçlar yalnızca veya temelde bir test faaliyetini desteklerken, bazıları birden fazla test faaliyetini destekleyebilir, ancak en yakından ilişkili oldukları faaliyet altında sınıflandırılır. Tek bir tedarikçinin sunduğu araçlar, özellikle birlikte çalışmak üzere tasarlanmış olanlar, entegre bir paket şeklinde sağlanabilir.

Bazı test araçları test sonucunu etkileyebilir cinste olabilir; bu, testin gerçekleşen çıktılarını etkileyebilecekleri anlamına gelmektedir. Örneğin, bir uygulama için gerçekleşen yanıt süreleri, bir performans test aracı tarafından yürütülen ekstra komutlar nedeniyle farklılaşabilir veya bir kapsam aracı kullanılması nedeniyle ulaşılan kod kapsamı miktarı hatalı olabilir. Bu tür araçların kullanımı sebebiyle sonucun değişmesine gözlemci etkisi denir. probe effect

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

Testlerin ve test yazılımının yönetimi için araç desteği

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

- Test yönetim araçları ve uygulama yaşam döngüsü yönetimi araçları (ALM) Application Lifecycle Management
- Gereksinim yönetim araçları (örneğin, test nesneleri arasında izlenebilirliğin sağlanması için)
- Hata yönetim araçları
- Yapılandırma yönetimi araçları
- Sürekli entegrasyon araçları (D)

Statik testler için araç desteği

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

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

Test tasarımı ve uyarlama için araç desteği

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

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

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

Test koşumu ve kayıt için araç desteği

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

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

Performans ölçümü ve dinamik analiz için araç desteği

performance and load testing

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

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

Özel test ihtiyacı için araç desteği

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

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

6.1.2 Test Otomasyonunun Faydaları ve Riskleri

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

Test koşumunu desteklemek için araç kullanmanın potansiyel faydaları aşağıda verilmiştir:

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

Testleri desteklemek için araç kullanmanın potansiyel riskleri aşağıda verilmiştir:

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

vendor

- Tedarikçi; bakım, destek, yeni versiyonlar ve hata düzeltmeleri için yeterli desteği vermeyebilir
- Açık kaynak kodlu bir yazılımın geliştirilmesi geçici olarak durdurulabilir
- Yeni bir platform veya teknoloji test aracı tarafından desteklenmeyebilir
- Aracın net bir sahibi olmayabilir (ör. güncellemeler için)

6.1.3 Test Koşumu ve Test Yönetim Araçlarına Dair Önemli Hususlar

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

Test koşumu araçları

Test koşumu araçları, otomatikleştirilmiş test betikleri kullanarak test nesnelerini koşturur. Bu tür bir araç, genellikle belirgin bir fayda elde etmek için önemli ölçüde çalışma gerektirir.

Capturing test approach (Görüntü yakalamaya dayali test yaklaşımını):

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

Data-driven test approach:

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

Keyword-driven test approach:

Aksiyon kelimesi güdümlü test yaklaşımında, genel bir betik, gerçekleştirilecek işlemleri tanımlayan anahtar kelimeleri (bunlara aksiyon kelimeleri de denir) işler, ardından ilişkili test verilerini işlemek için anahtar kelime betiklerini çağırır. Test uzmanları (betik diline hakim olmasalar bile), anahtar kelimeleri ve ilgili verileri kullanarak testleri tanımlayabilirler, bunlar da

test edilen uygulamaya göre uyarlanabilir. Veri güdümlü ve aksiyon kelimesi güdümlü test yaklaşımlarının ayrıntıları ve örnekleri, ISTQB-TAE İleri Seviye Test Otomasyon Mühendisi Ders Programı, Fewster 1999 ve Buwalda 2001'de verilmiştir.

Yukarıdaki yaklaşımlar, betik dilinde uzmanlaşmış birinin (test uzmanları, yazılımcılar veya test otomasyonu uzmanları) bulunmasını gerektirir. Kullanılan betik oluşturma tekniğinden bağımsız olarak, her test için beklenen sonuçların testten elde edilen gerçekleşen sonuçlarla karşılaştırılması gerekir; bu dinamik olarak (test koşturulurken) gerçekleştirilebilir veya daha sonra (koşum sonrası) karşılaştırma için sonuçlar saklanır.

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

Test yönetim araçları

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

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

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

6.2 Test Araçlarındaki Önemli Hususlar

6.2.1 Araç Seçiminde Ana Prensipler

Kurum için araç seçiminde dikkate alınması gereken ana hususlar şunlardır:

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

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

kavram ispatı = proof-of-concept

6.2.2 Bir Kurumda Bir Aracı Uygulamaya Koymak için Pilot Projeler

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

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

6.2.3 Araçlar için Başarı Faktörleri

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

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

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

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

7 Referanslar

Standartlar

ISO/IEC/IEEE 29119-1 (2013) Yazılım ve sistem mühendisliği - Yazılım testleri - Bölüm 1: Kavramlar ve tanımlar

ISO/IEC/IEEE 29119-2 (2013) Yazılım ve sistem mühendisliği - Yazılım testleri - Bölüm 2: Test süreçleri

ISO/IEC/IEEE 29119-3 (2013) Yazılım ve sistem mühendisliği - Yazılım testleri - Bölüm 3: Test dokümantasyonu

ISO/IEC/IEEE 29119-4 (2015) Yazılım ve sistem mühendisliği - Yazılım testleri - Bölüm 4: Test teknikleri

ISO/IEC 25010, (2011) Sistem ve yazılım mühendisliği - Sistem ve Yazılım Kalite Gereksinimleri ve Değerlendirme (SQuaRE) Sistem ve yazılım kalitesi modelleri

ISO/IEC 20246: (2017) Yazılım ve sistem mühendisliği - Çalışma ürünü gözden geçirmeleri

UML 2.5, Birleşik Modelleme Dili Başvuru Kılavuzu, http://www.omg.org/spec/UML/2.5.1/, 2017

ISTQB dokümanları

ISTQB Terimler Sözlüğü

ISTQB Temel Seviye Genel Bakış 2018

ISTQB-MBT Temel Seviye Model Bazlı Test Uzmanı Devam Kursu Ders Programı

ISTQB-AT Temel Seviye Çevik Test Uzmanı Devam Kursu Ders Programı

ISTQB-ATA İleri Seviye Test Analisti Ders Programı

ISTQB-ATM İleri Seviye Test Yöneticisi Ders Programı

ISTQB-SEC İleri Seviye Güvenlik Test Uzmanı Ders Programı

ISTQB-TAE İleri Seviye Test Otomasyon Mühendisi Ders Programı

ISTQB-ETM Uzman Seviye Test Yönetimi Ders Programı

ISTQB-EITP Uzman Seviye Test Sürecinin İyileştirilmesi Ders Programı

Kitaplar ve Makaleler

Beizer, B. (1990*) Yazılım Test Teknikleri (2. baskı)*, Van Nostrand Reinhold: Boston MA

Black, R. (2017) *Çevik Yazılım Testi Temelleri*, BCS Learning & Development Ltd: Swindon İngiltere

Black, R. (2009) *Test Sürecinin Yönetimi (3. baskı)*, John Wiley & Sons: New York NY

Buwalda, H. ve ark. (2001) *Entegre Test Tasarımı ve Otomasyonu*, Addison Wesley: Reading MA Copeland, L. (2004) *Yazılım Test Tasarımı için Kılavuz*, Artech House: Norwood MA Craig, R. ve Jaskiel, S. (2002) *Sistematik Yazılım Testleri*, Artech House: Norwood MA Crispin, L. ve Gregory, J. (2008) *Çevik Test*, Pearson Education: Boston MA

Fewster, M. ve Graham, D. (1999) *Yazılım Test Otomasyonu*, Addison Wesley: Harlow UK Gilb, T. ve Graham, D. (1993) *Yazılım Teftişi*, Addison Wesley: Reading MA

Graham, D. ve Fewster, M. (2012) *Test Otomasyon Deneyimleri*, Pearson Education: Boston MA Gregory, J. ve Crispin, L. (2015) *Çevik Testler Hakkında Daha Fazlası*, Pearson Education: Boston MA

Jorgensen, P. (2014) *Yazılım Testleri, Bir Zanaatkar Yaklaşımı (4. baskı)*, CRC Press: Boca Raton FL

Kaner, C., Bach, J. ve Pettichord, B. (2002) *Yazılım Testlerinden Alınan Dersler*, John Wiley & Sons: New York NY

Kaner, C., Padmanabhan, S. ve Hoffman, D. (2013) *Alan Testi Çalışma Kitabı*, Context-Driven Press: New York NY

Kramer, A., Legeard, B. (2016) *Model Bazlı Test Esasları: ISTQB Sertifikalı Model Bazlı Test Uzmanı için Rehber: Temel Seviye*, John Wiley & Sons: New York NY

Myers, G. (2011) *Yazılım Testi Sanatı*, (3. baskı), John Wiley & Sons: New York NY

Sauer, C. (2000) "Yazılım Geliştirmedeki Teknik Gözden Geçirmelerin Etkinliği: Davranış Güdümlü bir Araştırma Programı," *IEEE Transactions on Software Engineering, Cilt 26, Sayı 1, sayfa 1-*

Shull, F., Rus, I., Basili, V. Temmuz 2000. "Bakış Açısına Dayalı Okuma, Gereksinim Teftişleri Nasıl Geliştirebilir." *IEEE Computer, Cilt 33, Sayı 7, sayfa 73-79*

van Veenendaal, E. (ed.) (2004) *Test Pratisyeni* (Bölüm 8 - 10), UTN Publishers: Hollanda

Wiegers, K. (2002) *Yazılımda Eş-Gözden Geçirmeler*, Pearson Education: Boston MA

Weinberg, G. (2008) *Mükemmel Yazılım ve Testler hakkındaki Diğer Yanılsamalar*, Dorset House: New York NY

Diğer Kaynaklar (Bu Ders Programında doğrudan referans verilmeyen)

Black, R., van Veenendaal, E. ve Graham, D. (2012) *Yazılım Testinin Temelleri: ISTQB Sertifikasyonu (3. baskı)*, Cengage Learning: Londra İngiltere

Hetzel, W. (1993) *Yazılım Testleri için Tam Kılavuz (2. baskı)*, QED Information Sciences: Wellesley MA

Spillner, A., Linz, T. ve Schaefer, H. (2014) *Yazılım Testinin Temelleri (4. baskı)*, Rocky Nook: San Rafael CA

8 Ek A - Ders Programı Arka Planı

Bu Dokümanın Geçmişi

Bu doküman, ISTQB (www.istqb.org) tarafından onaylanan birinci seviye uluslararası yeterlilik programı olan ISTQB Sertifikalı Test Uzmanı Temel Seviye Ders Programıdır.

Bu doküman, 2014 ve 2018 yılları arasında International Software Testing Qualifications Board (ISTQB) tarafından atanan üyelerden oluşan bir çalışma grubu tarafından hazırlanmıştır. 2018 versiyonu, önce tüm ISTQB üye kurullarından temsilciler tarafından ve ardından uluslararası yazılım test topluluğundan seçilen temsilciler tarafından gözden geçirilmiştir.

Temel Seviye Sertifika Yeterliliğinin Hedefleri

- Yazılım testinin, yazılım mühendisliğinin temel ve profesyonel bir uzmanlığı olarak görülmesini sağlamak.
- Test uzmanlarının kariyerlerinin gelişimi için standart bir çerçeve sağlamak.
- Profesyonel test uzmanlarının işverenler, müşteriler ve meslektaşları tarafından tanınmasını sağlamak ve test uzmanlarının yetkinliklerini güçlendirmek.
- Tüm yazılım mühendisliği disiplinlerinde tutarlı ve iyi test uygulamalarını teşvik etmek.
- Sektörle alakalı ve değerli olan test konularını belirlemek.
- Yazılım tedarikçilerinin sertifikalı test uzmanları işe almalarına ve böylece test uzmanı işe alım politikalarını tanıtarak rakiplerine karşı ticari avantaj elde etmelerine olanak sağlamak.
- Bu konuda uluslararası kabul görmüş bir yeterlilik kazanabilmeleri için test uzmanları ve testlere ilgisi olanlar için bir fırsat sağlamak.

Uluslararası Yeterliliğin Hedefleri

- Farklı ülkelerdeki test bilgilerini karşılaştırma imkânı kazanmak.
- Test uzmanlarının yabancı ülkelerde çalışabilmelerini kolaylaştırmak.
- Çok uluslu/uluslararası projelerin test konularında ortak bir anlayış kazanmalarını sağlamak.
- Dünya çapında nitelikli test uzmanı sayısını artırmak.
- Herhangi bir ülkeye özgü olmaktan ziyade, uluslararası bazda geçerli olan etki ve değere sahip bir inisiyatif oluşturmak.

- Ders programı ve terminoloji ile test konusunda ortak bir uluslararası anlayış ve bilgi birliği geliştirmek ve tüm katılımcıların testler konusundaki bilgi seviyesini artırmak.
- Yazılım testlerini bir meslek olarak daha fazla ülkede tanıtmak.
- Test uzmanlarının kendi dillerinde tanınmış bir yeterlilik kazanmasını sağlamak.
- Bilgi ve kaynakların ülkeler arasında paylaşılmasını sağlamak.
- Test uzmanlarının ve yeterlilik programının uluslararası tanınırlığını sağlamak.

Bu Yeterlilik için Giriş Gereksinimleri

ISTQB Sertifikalı Test Uzmanı Temel Seviye sınavına girmek için herhangi bir giriş kriteri yoktur, yazılım testlerine ilgi duyan veya bu alanda kariyer yapmak isteyen herkes bu sınava girebilir. Ancak, adayların aşağıdaki vasıflara sahip olması da önemle önerilir:

- Yazılım geliştirme veya yazılım testlerinde en azından asgari bir altyapıya sahip olmak: örneğin bir sistem veya kullanıcı kabul testi uzmanı veya yazılım geliştirici olarak altı aylık deneyim
- ISTQB tarafından tanınan üye kurullarından birinin ISTQB standartlarına göre akredite ettiği bir kursa katılmış olmak

Yazılım Testlerinde Temel Seviye Sertifikasının Arka Planı ve Geçmişi

Yazılım testi uzmanlarının bağımsız sertifikasyonu, İngiltere'de, 1998'de bir yazılım test kurulu oluşturulmasıyla, British Computer Society's Information Systems Examination Board (ISEB), başladı (www.bcs.org.uk/iseb). 2002'de, Almanya'daki ASQF bir Alman test uzmanı yeterlilik programını (www.asqf.de) desteklemeye başladı. Bu ders programı, ISEB ve ASQF ders programlarına dayanmaktadır; yeniden düzenlenmiş, güncellenmiş ve ek içeriğe sahiptir ve test uzmanlarına en pratik desteği sağlayacak konulara odaklanmıştır.

Bu uluslararası sertifika yayımlanmadan önce verilen Yazılım Testinde Temel Seviye Sertifikası (örneğin ISEB, ASQF veya ISTQB tarafından tanınan bir üye kuruldan), uluslararası sertifikaya eşdeğer kabul edilir. Temel Seviye Sertifikası'nın son geçerlilik süresi yoktur ve yenilenmesi gerekmez. Verildiği tarih, sertifika üzerinde belirtilmiştir.

Her katılımcı ülkede, yerel konular, ISTQB tarafından tanınan ulusal veya bölgesel bir Yazılım Test Kurulu tarafından kontrol edilir. Üye kurulların görevleri ISTQB tarafından belirlenir, ancak her ülkenin kendi içinde yürürlüğe alınır. Ülke kurullarının görevlerinin arasında, eğitim kurumlarının akreditasyonu ve ISTQB sınavlarının düzenlenmesi yer alır.

9 Ek B - Öğrenme Hedefleri/Bilginin Bilişsel Seviyesi

Bu ders programı aşağıdaki öğrenme hedeflerini içerir. Ders programındaki her konu, ilgili öğrenme hedefine göre incelenecektir.

Seviye 1: Hatırlama (K1)

Aday terim ve kavramları tanımalı ve hatırlamalıdır.

Anahtar kelimeler: Tanımlama, hatırlama, unutmama, tanıma, bilme

Örnekler:

failure "Arıza"nın tanımının aşağıdaki gibi olduğunu bilmelidir:

- "Son kullanıcıya veya başka bir paydaşa hizmetin teslim edilmemesi" veya
- "Birim veya sistemin beklenen veriminden, servisinden veya sonucundan sapması"

Seviye 2: Anlama (K2)

Aday, konuyla ilgili ifadelerin nedenlerini veya açıklamalarını seçebilir ve test kavramı için özetleme yapabilir, karşılaştırabilir, sınıflandırabilir, kategorilere ayırabilir ve örnekler verebilir.

Anahtar kelimeler: Özetle, genelleştir, soyutla, sınıflandır, karşılaştır, eşle, zıddıyla karşılaştır, örnekle, yorumla, tercüme et, temsil et, anlam çıkar, sonuçlandır, kategorize et, model kur

Örnekler:

Test analizi ve tasarımın neden olabildiğince erken gerçekleşmesi gerektiğini açıklayabilir:

- Çözülmeleri daha düşük maliyetliyken hataları bulmak için
- Önce en önemli hataları bulmak için

Entegrasyon ve sistem testleri arasındaki benzerlik ve farkları açıklayabilir:

- Benzerlikler: Entegrasyon testleri ve sistem testleri için test nesneleri birden fazla birim içerir ve hem entegrasyon testleri hem de sistem testleri fonksiyonel olmayan test türlerini içerebilir.
- Farklar: Entegrasyon testleri arayüzler ve etkileşimlere odaklanır; sistem testleri, sürecin uçtan uca işletilmesi gibi tüm sistem yönlerine odaklanır.

Seviye 3: Uygulama (K3)

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

Anahtar kelimeler: Uygula, yürüt, kullan, prosedüre uy, prosedürü uygula

Örnekler:

- Geçerli ve geçersiz paylar için sınır değerlerini belirleyebilir.
- Tüm geçişleri kapsayacak şekilde belirli bir durum geçişi diyagramından test senaryolarını seçebilir.

Referans (Öğrenme hedeflerinin bilişsel düzeyleri için)

Anderson, L. W. ve Krathwohl, D. R. (eds) (2001) *Öğrenme, Öğretme ve Denetleme için bir Sınıflandırma: Bloom'un Eğitim Hedefleri Sınıflandırmasının Gözden Geçirilmesi*, Allyn & Bacon: Boston MA

10 Ek C - Sürüm Notları

ISTQB Temel Seviye Ders Programı 2018, 2011 sürümü üzerinden büyük ölçüde güncellenmiş ve yeniden yazılmıştır. Bu nedenle, kısım ve bölüme göre ayrıntılı sürüm notları yoktur. Ancak, ana değişikliklerin bir özeti burada verilmektedir. Ek olarak, ayrı bir Sürüm Notları dokümanında ISTQB, Temel Seviye Ders Programının 2011 versiyonundaki öğrenme hedefleri ile Temel Seviye Ders Programının 2018 versiyonundaki öğrenme hedefleri arasında, hangilerinin eklendiğini, güncellendiğini veya çıkarıldığını göstererek izlenebilirlik sağlamıştır.

2017 yılı itibariyle, 100'den fazla ülkede 550.000'den fazla kişi temel seviye sınavına girmiş ve 500.000'den fazla dünya çapında sertifikalı test uzmanı olmuştur. Her bir katılımcının sınavı geçebilmek için Temel Seviye Ders Programını okuduğu düşünülürse bu, Temel Seviye Ders Programını muhtemelen şimdiye kadar en çok okunmuş yazılım test dokümanı yapmaktadır!

Bu büyük güncelleme, bu birikime ilişkin olarak ve ISTQB'nin küresel test topluluğunda bir sonraki 500.000 kişiye sağlayacağı değeri artırmak için yapılmıştır.

Bu versiyonda tüm öğrenme hedefleri, olabildiğince küçültülmek, her öğrenme hedefinden o öğrenme hedefiyle ilgili içerik bölümlerine (ve sınav sorularına) net bir izlenebilirlik oluşturmak ve aynı şekilde içerik bölümlerinden (ve sınav sorularından) ilgili öğrenme hedefine net bir izlenebilirlik oluşturmak için düzenlenmiştir. Ek olarak, diğer ISTQB ders programlarında kullanılan kanıtlanmış sezgisel yöntemler ve formüller kullanılarak, bölümler için ayrılan ve her bölümde ele alınacak olan öğrenme hedeflerinin analizine dayanan süreler, ders programının 2011 versiyonundakinden daha gerçekçi hale getirilmiştir.

Bu her ne kadar temel bir ders programı olsa da, zamanın en iyi uygulamalarına ve tekniklerine uyum sağlamak amacıyla, özellikle yazılım geliştirme yöntemleri (ör. Scrum ve sürekli dağıtım) ve teknolojileri (ör. nesnelerin interneti) açısından ders materyalinin sunumunu modernize etmek için değişiklikler yapılmıştır. Referans verilen standartlar daha güncel hale getirilmek için aşağıdaki gibi güncellenmiştir:

1. ISO/IEC/IEEE 29119, IEEE Standardı 829'un yerini almıştır
2. ISO/IEC 25010, ISO 9126'nın yerini almıştır
3. ISO/IEC 20246, IEEE 1028'in yerini almıştır

Ek olarak, ISTQB dokümanları son on yılda muazzam bir şekilde arttığı için, gereken yerlerde diğer ISTQB ders

programlarındaki ilgili materyallere kapsamlı çapraz referanslar eklenmiş ve tüm ders programları ve ISTQB Terimler Sözlüğü ile uyum için dikkatlice gözden geçirilmiştir. Amaç, pratik kullanım ve bilgi ve beceriler arasındaki dengeyi artırmaya odaklanarak bu versiyonun okunmasını, anlaşılmasını, öğrenilmesini ve tercüme edilmesini kolaylaştırmaktır.

Bu sürümde yapılan değişikliklerin ayrıntılı analizi için ISTQB Sertifikalı Test Uzmanı Temel Seviye Genel Bakış 2018 dokümanını inceleyiniz.

11 İndeks