Sertifikalı Test Uzmanı

Temel Seviye Ders Programı

v4.0

International Software Testing Qualifications Board

Telif Hakkı Bildirimi

Revizyon Geçmişi

Versiyon	Tarih	Notlar
CTFL v4.0	21.04.2023	CTFL v4.0 – Genel yayın versiyonu
CTFL v3.1.1	01.07.2021	CTFL v3.1.1 – Telif hakkı ve logo güncellemesi
CTFL v3.1	11.11.2019	CTFL v3.1 – Minör güncellemeler içeren bakım sürümü
ISTQB 2018	27.04.2018	CTFL v3.0 – Aday genel yayın versiyonu
ISTQB 2011	1.04.2011	CTFL Ders Programı Bakım Sürümü
ISTQB 2010	30.03.2010	CTFL Ders Programı Bakım Sürümü
ISTQB 2007	01.05.2007	CTFL Ders Programı Bakım Sürümü
ISTQB 2005	01.07.2005	Sertifikalı Test Uzmanı Temel Seviye Ders Programı v1.0
ASQF V2.2	07.2003	ASQF Ders Programı Temel Seviye Versiyon v2.2 "Yazılım Testleri için Temel Seviye Ders Programı"
ISEB V2.0	25.02.1999	ISEB Yazılım Testlerinin Temelleri Ders Programı v2.0

Önsöz

ISTQB® Sertifikalı Test Uzmanı Temel Seviye Ders Programının 2018 Türkçe versiyonunun yayınlamasının üzerinden yaklaşık 5 yıl geçti. Bu süre zarfında hem yazılım test sektöründe çok önemli gelişmeler oldu hem de sektör Türkiye'de ve dünyada yıllık ortalama %20 oranında bir ivmeyle büyüme kaydetti. Haziran 2023 itibariyle Türkiye'de ISTQB® sınavlarına giren katılımcı sayısı 10.000'i geçmiş, 5.000'den fazla katılımcı da sınavlarda başarılı olarak ISTQB® Uluslararası Sertifikalı Test Uzmanı sertifikalarını almaya hak kazanmıştır. Bu rakamlar dünya genelinde 1.3 milyondan fazla sınav ve 914.000'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, Nisan 2023'de bilişim sektörünün hizmetine sunulmuştur. Dokümanda özellikle Çevik yazılım geliştirme ve DevOps yaklaşımlarının teste etkisi göze çarpmaktadır. Yazılım Test ve Kalite Derneği çeviri çalışma grubu olarak bizler de bu değerli dokümanı sektör profesyonellerine sunmak amacıyla 2024 yılı içerisinde çalışmalarımıza başlayıp çeviriyi Mart 2024 itibariyle bitirmiş bulunmaktayız.

Daha önceki versiyonların 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

Mart, 2024

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
Abdurrahman Karabacak
Ahmet Esat Genç
Alieren Dasdemir
Alp Ekin
Alparslan Yiğid
Alper Turp
Barış Küçük
Burak Yakupoğlu
Burcu Özel
Çiğdem Aldan
Didem Çolak Arslan
Ece Kılıç
Ecem Güney
Elif Göktaş
Elif Yağlıkçı
Esra Karaarslan
Gençay Genç
Gülsüm Güngör
Hakan Güvez
Hasan Özyer
İbrahim Seyfullah Babaarslan
İdil Bilginturan
Kader Çelik
Kadir Tepecik
Kağan Hazal Kocdemi
L. Koray Yitmen
Mahir Aldağ
Mehmet Kıvanç Bayram
Meltem Aydaş
Mert Çalışkan
Muhammet Topcu
Mustafa Aydın
Nergiz Gençaslan
Salim öncü
Serkan Cura
Zeynep Teke

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 10.000'den fazla 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 Temel Seviye 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
- ISTQB® Uluslararası Sertifikalı Performans Testi Sınavı
- ISTQB® Uluslararası Sertifikalı Test Otomasyon Mühendisi Sınavı
- ISTQB® Uluslararası Sertifikalı Yapay Zeka Sınavı
- ISTQB® Uluslararası Sertifikalı Mobil Uygulama Sınavı
- ISTQB® Uluslararası Sertifikalı Otomotiv Yazılım Sınavı

Uluslararası TestIstanbul Konferansları – www.testistanbul.org

Dernek, 2010 yılından bu yana Uluslararası TestIstanbul Konferanslarını düzenlemektedir. Geçtiğimiz 14 konferansta 50'den fazla ülkeden, 70'den fazla konuşmacı ve 7.000'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,
- Testİzmir,
- Test 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 v4.0
- 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
- Bir Ejderhadan Yazılım Test Dersi
- Çevik Bir Dünyada TMMi

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. Mart 2024 itibariyle burs sağlanan toplam bursiyer sayısı 87'e ulaşmıştır.

İçindekiler Tablosu

Teşekkür

Bu doküman ISTQB® Genel Kurulu tarafından 21 Nisan 2023'te resmi olarak yayınlanmıştır.

Doküman, ISTQB Temel Seviye ve Çevik Çalışma Gruplarından bir ekip tarafından hazırlanmıştır: Laura Albert, Renzo Cerquozzi (Başkan Yardımcısı), Wim Decoutere, Klaudia Dussa-Zieger, Chintaka Indikadahena, Arnika Hryszko, Martin Klonk, Kenji Onishi, Michaël Pilaeten (Eş Başkan), Meile Posthuma, Gandhinee Rajkomar, Stuart Reid, Eric Riou du Cosquer (Eş Başkan), Jean-François Riverin, Adam Roman, Lucjan Stapp, Stephanie Ulrich (Başkan Yardımcısı), Eshraka Zakaria.

Ekip, teknik gözden geçirme için Stuart Reid, Patricia McQuaid ve Leanne Howard ile önerileri ve girdileri için gözden geçirme ekibi ve Üye Kurullara teşekkür eder.

Bu ders programının gözden geçirilmesi, yorumlanması ve oylanmasında aşağıdaki kişiler yer almıştır: Adam Roman, Adam Scierski, Ágota Horváth, Ainsley Rood, Ale Rebon Portillo, Alessandro Collino, Alexander Alexandrov, Amanda Logue, Ana Ochoa, André Baumann, André Verschelling, Andreas Spillner, Anna Miazek, Armin Born, Arnd Pehl, Arne Becher, Attila Gyúri, Attila Kovács, Beata Karpinska, Benjamin Timmermans, Blair Mo, Carsten Weise, Chinthaka Indikadahena, Chris Van Bael, Ciaran O'Leary, Claude Zhang, Cristina Sobrero, Dandan Zheng, Dani Almog, Daniel Säther, Daniel van der Zwan, Danilo Magli, Darvay Tamás Béla, Dawn Haynes, Dena Pauletti, Dénes Medzihradszky, Doris Dötzer, Dot Graham, Edward Weller, Erhardt Wunderlich, Eric Riou Du Cosquer, Florian Fieber, Fran O'Hara, François Vaillancourt, Frans Dijkman, Gabriele Haller, Gary Mogyorodi, Georg Sehl, Géza Bujdosó, Giancarlo Tomasig, Giorgio Pisani, Gustavo Márquez Sosa, Helmut Pichler, Hongbao Zhai, Horst Pohlmann, Ignacio Trejos, Ilia Kulakov, Ine Lutterman, Ingvar Nordström, Iosif Itkin, Jamie Mitchell, Jan Giesen,
Jean-Francois Riverin, Joanna Kazun, Joanne Tremblay, Joëlle Genois, Johan Klintin, John Kurowski, Jörn Münzel, Judy McKay, Jürgen Beniermann, Karol Frühauf, Katalin Balla, Kevin Kooh, Klaudia Dussa-Zieger, Klaus Erlenbach, Klaus Olsen, Krisztián Miskó, Laura Albert, Liang Ren, Lijuan Wang, Lloyd Roden, Lucjan Stapp, Mahmoud Khalaili, Marek Majernik, Maria Clara Choucair, Mark Rutz, Markus Niehammer, Martin Klonk, Márton Siska, Matthew Gregg, Matthias Hamburg, Mattijs Kemmink, Maud Schlich, May Abu-Sbeit, Meile Posthuma, Mette Bruhn-Pedersen, Michal Tal, Michel Boies, Mike Smith, Miroslav Renda, Mohsen Ekssir, Monika Stocklein Olsen, Murian Song, Nicola De Rosa, Nikita Kalyani, Nishan Portoyan, Nitzan Goldenberg, Ole Chr. Hansen, Patricia McQuaid, Patricia Osorio, Paul Weymouth, Pawel Kwasik, Peter Zimmerer, Petr Neugebauer, Piet de Roo, Radoslaw Smilgin, Ralf Bongard, Ralf Reissing, Randall Rice, Rik Marselis, Rogier Ammerlaan, Sabine Gschwandtner, Sabine Uhde, Salinda Wickramasinghe, Salvatore Reale, Sammy Kolluru, Samuel Ouko, Stephanie Ulrich, Stuart Reid, Surabhi Bellani, Szilard Szell, Tamás Gergely, Tamás Horváth, Tatiana Sergeeva, Tauhida Parveen, Thaer Mustafa, Thomas Eisbrenner, Thomas Harms, Thomas Heller, Tobias Letzkus, Tomas Rosenqvist, Werner Lieblang, Yaron Tsubery, Zhenlei Zuo ve Zsolt Hargitai.

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

ISTQB Temel Seviye Çalışma Grubu (Basım 2011): Thomas Müller (Başkan), Debra Friedenberg. Çekirdek ekip, ders programının mevcut versiyonuna yönelik 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.

ISTQB Temel Seviye Çalışma Grubu (Basım 2010): Thomas Müller (Başkan), Rahul Verma, Martin Klonk ve Armin Beer. Çekirdek ekip, önerileri için gözden geçirme ekibine (Rex Black, Mette Bruhn-Pederson, Debra Friedenberg, Klaus Olsen, Judy McKay, Tuula Pääkkönen, Meile Posthuma, Hans Schaefer, Stephanie Ulrich, Pete Williams, Erik van Veenendaal) ve tüm Üye Kurullara teşekkür eder.

ISTQB 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.

ISTQB 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 referansları değiştirebilir.
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.

0.2. Yazılım Testlerinde 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, yazılımcı ve geliştirme ekibi üyeleri gibi çeşitli rollerdeki kişiler dahildir. Bu ders programı ayrıca proje yöneticileri, kalite yöneticileri, ürün sahipleri, 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.

0.3. Test Uzmanları için Kariyer Yolu

ISTQB® programı tam kapsamlı bir bilgi birikimi sunarak kariyerlerinin tüm aşamalarında test uzmanlarına destek sağlar. ISTQB® Temel Seviye sertifikasını alan kişiler Çekirdek İleri Seviyeler (Test Analisti, Teknik Test Analisti ve Test Yöneticisi) ve dolayısıyla Uzman Seviyesi (Test Yönetimi veya Test Sürecinin İyileştirilmesi) konularıyla da ilgilenebilirler. Çevik bir ortamda test uygulamaları ile ilgili becerilerini geliştirmek isteyenler Çevik Teknik Test Uzmanlığı ya da Ölçek Bazında Çevik Test Liderliği sertifika programlarını düşünebilirler. Uzman müfredatı, spesifik test yaklaşımları ve test aktivitelerine yer verilen (ör. test otomasyonu, yapay zeka (YZ) testi, model bazlı test, mobil uygulama testlerinde), spesifik test alanlarıyla ilgili (ör. performans testi, kullanılabilirlik testi, kabul testi, güvenlik testi) veya belirli sektörler için (ör. otomotiv veya oyun) test bilgilerini bir araya getiren alanlara ilişkin derinlemesine bir inceleme sunar. ISTQB'nin Sertifikalı Test Uzmanı Programı hakkında en güncel bilgiler için www.istqb.org adresini ziyaret edin.

0.4. İş Çıktıları

Bu bölümde, Temel Seviye sertifikasyonu alan bir kişiden beklenen 14 iş çıktısına yer verilmiştir.

Temel Seviye Sertifikalı Test Uzmanları...

FL-BO1	Testin ne olduğunu ve neden yararlı olduğunu anlayabilir
FL-BO2	Yazılım testinin temel kavramlarını anlayabilir
FL-BO3	Testin bağlamına göre uygulanacak test yaklaşımını ve aktiviteleri tanımlayabilir
FL-BO4	Dokümantasyonun kalitesini değerlendirebilir ve iyileştirebilir
FL-BO5	Testin etkinliğini ve verimliliğini artırabilir
FL-BO6	Test sürecini yazılım geliştirme yaşam döngüsüyle uyumlu hale getirebilir
FL-BO7	Test yönetimi prensiplerini anlayabilir
FL-BO8	Net ve anlaşılır hata raporları yazabilir ve bunları aktarabilir
FL-BO9	Testle ilgili öncelik ve eforu etkileyen faktörleri anlayabilir
FL-BO10	Disiplinler arası bir ekibin üyesi olarak çalışabilir
FL-BO11	Test otomasyonuyla ilgili riskleri ve faydaları bilebilir
FL-BO12	Test için gerekli olan becerileri tanımlayabilir
FL-BO13	Riskin test üzerindeki etkisini anlayabilir
FL-BO14	Testin ilerleme durumu ve test kalitesi hakkında etkin bir rapor sunabilir

0.5. Sınav Kapsamındaki Öğrenme Hedefleri ve Bilginin Bilişsel Seviyesi

Öğrenme hedefleri iş hayatına yönelik çıktıları destekler ve Sertifikalı Test Uzmanı Temel Seviye sınavlarını hazırlamakta kullanılır. Genel olarak, bu ders programının 1-6. konularının tüm içeriği, K1 seviyesinde sınav kapsamındadır. Adaydan altı konudan herhangi birinde belirtilen bir anahtar kelimeyi veya kavramı bilmesi veya hatırlaması istenebilir. Özel öğrenme hedefi seviyeleri, her konunun başında gösterilmektedir ve aşağıdaki gibi sınıflandırılmıştır:

- K1: Hatırla
- K2: Anla
- K3: Uygula

Öğrenme hedeflerine ilişkin diğer detaylar ve örneklere Ek A'da yer verilmiştir. Konu başlıklarının hemen altında anahtar kelimeler olarak listelenen tüm terimlerin tanımlarının öğrenme hedeflerinde açıkça belirtilmiş olmasa bile hatırlanması gerekmektedir (K1).

0.6. 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 ve kitaplar referans olarak dahil edilmiştir (Konu 7); ancak bunların içerikleri, bu ders programında bahsedildiği kadar sınava dahildir. Bahsedilmeyen kısım ve içerikler sınav kapsamına dahil değildir. Temel Seviye Sınav Yapıları ve Kuralları belgesine bakın.

0.7. 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. Bu ders programının akreditasyon yönergeleri Süreç Yönetimi ve Uyumluluk Çalışma Grubu tarafından yayınlanan genel Akreditasyon Yönergelerine uygundur.

0.8. Standartların Ele Alınması

Temel Seviye Ders Programında bazı standartlar referans gösterilmiştir (ör. IEEE veya ISO standartları). Bu referanslar bir çerçeve sunar (kalite karakteristiğine ilişkin ISO 25010 standardına dair referanslarda olduğu gibi) veya okuyucu tarafından istenirse bir ek bilgi kaynağı sunar. Standart belgeleri sınav kapsamında değildir. Standartlar hakkında daha fazla bilgi için konu 7'ye bakın.

0.9. Güncel Kalmak

Yazılım sektörü hızla değişiyor. ISTQB çalışma grupları, bu değişiklikleri ele almak ve paydaşların ilgili ve güncel bilgilere erişmesini sağlamak için www.istqb.org web sitesinde destek belgelere ve standartlardaki değişikliklere ilişkin bağlantılar oluşturdular. Bu bilgiler Temel Seviye ders programı kapsamında sınava tabi değildir.

0.10. Ayrıntı Düzeyi

Bu ders programındaki ayrıntı düzeyi, uluslararası düzeyde tutarlı 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 (anahtar kelimeler) listesi
- Her bilgi alanı için, ulaşılacak bilişsel öğrenme çıktılarını tanımlayan öğrenme hedefleri
- Genel kabul görmüş kaynaklara verilen referanslar dahil olmak üzere temel kavramların açıklaması

Ders programı içeriği, yazılım testlerindeki tüm bilgi birikiminin bir açıklaması değildir; sadece temel seviye eğitim kurslarında ele alınacak ayrıntı düzeyini gösterir. Kullanılan yazılım geliştirme yaşam döngüsünden (YGYD) bağımsız olarak tüm yazılım projelerine uygulanabilecek test kavramları ve

tekniklerine odaklanır.

0.11. Bu Ders Programı Nasıl Düzenlendi

Ders programı sınava tabi olan altı konudan oluşmaktadır. Her konuda yer alan en üstteki başlık o konu için ayrılan eğitim süresini belirtir. Zaman planlaması o konunun geneli için verilmiştir, daha detayda bir planlama verilmemiştir. Akredite eğitim kursları için ders programı aşağıdaki gibi altı konuya ayrılmış olup toplamda en az 1135 dakika (18 saat 55 dakika) kurs alınmasını gerektirir:

- Konu 1: Yazılım Testinin Temelleri (180 dakika)
    - Öğrenci, testle ilgili temel prensipleri, testin neden gerekli olduğunu ve test hedeflerinin ne olduğunu öğrenir.
    - Öğrenci, test sürecini, önemli test aktivitelerini ve test yazılımını anlar.
    - Öğrenci, test için gereken temel yetenekleri anlar.
- Konu 2: Yazılım Geliştirme Yaşam Döngüsü Boyunca Test (130 dakika)
    - Öğrenci, test etmenin farklı yazılım geliştirme yaklaşımlarına nasıl dahil edildiğini öğrenir.
    - Öğrenci, önce-test-et yaklaşımları ve DevOps konseptlerini öğrenir.
    - Öğrenci, farklı test seviyelerini, test çeşitlerini ve bakım testini öğrenir.
- Konu 3: Statik Testler (80 dakika)
    - Öğrenci, statik test temellerini, geri bildirim ve gözden geçirme sürecini öğrenir.
- Konu 4: Test Analizi ve Tasarımı (390 dakika)
    - Öğrenci, çeşitli yazılım çalışma ürünlerinden test senaryoları elde etmek için kara kutu, beyaz kutu ve tecrübeye dayalı test tekniklerini nasıl uygulayacağını öğrenir.
    - Öğrenci, iş birliğine dayalı test yaklaşımını öğrenir.
- Konu 5: Test Aktivitelerini Yönetme (335 dakika)
    - Öğrenci, genel anlamda testleri nasıl planlayacağını ve test eforunu nasıl tahmin edebileceğini öğrenir.
    - Öğrenci, risklerin test kapsamını nasıl etkileyebileceğini öğrenir.
    - Öğrenci, test aktivitelerini nasıl izleyip kontrol edebileceğini öğrenir.
    - Öğrenci, yapılandırma yönetiminin testleri nasıl desteklediğini öğrenir.
    - Öğrenci, hataları net ve anlaşılır şekilde nasıl rapor edebileceğini öğrenir.
- Konu 6: Test Araçları (20 dakika)
    - Öğrenci, araçları sınıflandırmayı ve test otomasyonuna ilişkin risk ve faydaları anlamayı öğrenir.

1. Yazılım Testinin Temelleri – 180 dakika

Anahtar kelimeler

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 verisi, test tasarımı, test koşumu, test uyarlama, test gözetimi, test nesnesi, test hedefi, test planlama, test prosedürü, test sonucu, test etme, test yazılımı, sağlama, doğrulama

Konu 1 Öğrenme Hedefleri:

1.1 Yazılım Testi Nedir?

FL-1.1.1 (K1) Genel test hedeflerini tanımlamak

FL-1.1.2 (K2) Test ile hata ayıklamanın farklarını belirtmek

1.2 Yazılım Testi Neden Gereklidir?

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

FL-1.2.2 (K1) Yazılım testi ve kalite güvence arasındaki ilişkiyi anımsamak

FL-1.2.3 (K2) Kök neden, insan hatası, hata ve arıza arasındaki farkı ayırt etmek

1.3 Test Prensipleri

FL-1.3.1 (K2) Yedi test prensibini açıklamak

1.4 Test Aktiviteleri, Test Yazılımı ve Test Rolleri

FL-1.4.1 (K2) Farklı test aktivitelerini ve görevlerini özetlemek

FL-1.4.2 (K2) Proje bağlamının test süreci üzerindeki etkisini açıklamak

FL-1.4.3 (K2) Test aktivitelerini destekleyen test yazılımlarını ayırt etmek

FL-1.4.4 (K2) İzlenebilirliğin sağlanmasının önemini açıklamak

FL-1.4.5 (K2) Test etme sürecindeki farklı rolleri karşılaştırmak

1.5 Test Etme Sürecinde Gerekli Beceriler ve İyi Uygulamalar

FL-1.5.1 (K2) Test etme sürecinde gerekli genel becerilere örnekler vermek

FL-1.5.2 (K1) Tüm ekip yaklaşımının avantajlarını hatırlamak

FL-1.5.3 (K2) Testin bağımsızlığının yararlarını ve sakıncalarını açıklamak

1.1. Yazılım Testi Nedir?

Yazılımlar günlük hayatımızın ayrılmaz bir parçasıdır. Çoğu insan, beklendiği gibi çalışmayan bir yazılım ile karşılaşmıştır. Düzgün çalışmayan yazılımlar; para, zaman veya ticari itibar kaybetme ve hatta en uç durumlarda yaralanma veya ölüm gibi birçok soruna yol açabilir.
Yazılım testi yazılım kalitesini değerlendirir ve yazılımın kullanımı sırasında oluşabilecek yazılım hatası riskini azaltmaya yardımcı olur.

Yazılım testi, hata bulmak ve yazılım geliştirme sırasında üretilen eserlerin kalitesini değerlendirmek için yapılan bir aktiviteler bütünüdür. Bu eserlere, test edilirken test nesnesi adı verilir. Yazılım testi hakkındaki yaygın yanlış kanılardan biri yazılım testinin yazılımı çalıştırmaktan ve test sonuçlarını kontrol etmekten ibaret olduğudur. Ancak yazılım testi aynı zamanda başka aktiviteleri de içerir ve yazılım geliştirme yaşam döngüsüyle uyumlu olmalıdır (bkz. konu 2).

Test hakkındaki bir diğer yanlış kanı da test etme sürecinin tamamen test nesnesinin doğrulanmasına odaklı olduğudur. Her ne kadar yazılım testleri, doğrulamayı, diğer bir deyişle sistemin belirli gereksinimleri karşılayıp karşılamadığını kontrol etmeyi içerse de aynı zamanda sistemin operasyonel ortamda/ortamlarda kullanıcı ve diğer paydaş ihtiyaçlarını karşılayıp karşılamadığını kontrol etmek anlamına gelen sağlama yapmayı da içerir.

Yazılım testleri dinamik veya statik olabilir. Dinamik test içerisinde yazılımın çalıştırılması yer alırken statik testte bu yoktur. Statik test, gözden geçirmeleri (bkz. konu 3) ve statik analizi içerir. Dinamik test, farklı test teknikleri ve yaklaşımları kullanarak test senaryolarını elde eder (bkz. konu 4).

Yazılım testi sadece teknik bir aktivite değildir. Aynı zamanda uygun şekilde planlanması, yönetilmesi, tahmin edilmesi, izlenmesi ve kontrol edilmesi gerekir (bkz. konu 5).

Test uzmanları araçlar kullanır (bkz. konu 6) ancak yazılım testinin büyük ölçüde düşünsel bir aktivite olduğu ve test uzmanının uzmanlık bilgisine sahip olması, analitik beceriler kullanması ve eleştirel ve sistemsel düşünme yaklaşımını uygulaması gerektiği unutulmamalıdır (Myers 2011, Roman 2018).

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

1.1.1. Test Hedefleri

Genel test hedefleri:

- Gereksinimler, kullanıcı hikayeleri, tasarımlar ve kod gibi çalışma ürünlerini değerlendirmek
- Arızaları tetiklemek ve hataları bulmak
- Test nesnesinin gerekli kapsamını sağlamak
- Yazılımın kalite kriterlerini karşılayamama riskini azaltmak
- Belirtilen gereksinimlerin yerine getirilip getirilmediğini doğrulamak
- Test nesnesinin sözleşmeden kaynaklanan, yasal veya düzenleyici gereksinimlere uyumluluğunu doğrulamak
- Sağlıklı kararlar almalarını sağlamak için paydaşlara yeterli bilgiyi sunmak
- Test nesnesinin kalitesi hakkında güven oluşturmak
- Test nesnesinin eksiksiz olup olmadığının ve paydaşların beklediği şekilde çalıştığının sağlamasını yapmak

Test hedefleri; test edilen çalışma ürünü, test seviyesi, riskler ve takip edilen yazılım geliştirme yaşam döngüsünü (YGYD) içeren bağlama ve kurumsal yapı, rekabetçi hususlar veya pazara sürüm süresi gibi ticari bağlama ilişkin faktörlere bağlı olarak değişebilir.

1.1.2. Yazılım Testi ve Hata Ayıklama

Test etme ve hata ayıklama birbirinden farklı aktivitelerdir. Test etme süreci yazılımdaki hataların neden olduğu arızaları tetikleyebilir (dinamik test) veya test nesnesindeki hataları doğrudan bulabilir (statik test).

Dinamik test (bkz. konu 4) bir arızayı tetiklediğinde hata ayıklama işlemi bu arızanın (hataların) nedenlerinin bulunması, bu nedenlerin analiz edilmesi ve giderilmesiyle ilgilenir. Bu durumda tipik hata ayıklama süreci şunları içerir:

- Arızanın yeniden oluşturulması
- Teşhis (kök nedenin bulunması)
- Nedenin düzeltilmesi

Sonrasında gerçekleştirilen onaylama testleri düzeltmelerin problemi giderip gidermediğini kontrol eder. Tercihen onaylama testi başlangıçtaki testi gerçekleştiren aynı kişi tarafından yapılır. Düzeltmelerin test nesnesinin diğer bölümlerinde arızalara neden olup olmadığını kontrol etmek için daha sonra bir regresyon testi de yapılabilir (onaylama ve regresyon testine ilişkin daha detaylı bilgi için bkz. bölüm 2.2.3).

Statik test bir hata bulduğunda hata ayıklama işlemi bu hatayı gidermekle ilgilenir. Statik test hataları doğrudan bulduğundan ve arızalara neden olamayacağından yeniden oluşturmaya veya teşhise gerek yoktur (bkz. konu 3).

1.2. Yazılım Testi Neden Gereklidir?

Bir tür kalite kontrolü olarak test etme süreci belirlenen kapsam, zaman, kalite ve bütçe kısıtları dahilinde önceden kararlaştırılmış hedeflere ulaşılmasına yardımcı olur. Testin başarıya olan katkısı test ekibi aktiviteleriyle sınırlandırılmamalıdır. Her paydaş kendi test becerilerini kullanarak projeyi başarıya daha da yaklaştırabilir. Test bileşenleri, sistemleri ve ilgili dokümantasyon, yazılımdaki hataların belirlenmesine yardımcı olur.

1.2.1. Yazılım Testinin Başarıya Katkısı

Test süreci, hataların bulunmasına ilişkin maliyet etkin bir yol sunar. Daha sonra bu hatalar, test sürecinin daha yüksek kaliteli test nesnelerine katkıda bulunması için (test dışı bir aktivite olan hata ayıklama yoluyla) giderilebilir.

Test etme süreci, YGYD'nün çeşitli aşamalarında test nesnesinin kalitesini doğrudan değerlendirme imkanı sunar. Bu ölçümler, daha geniş bir proje yönetimi kapsamında kullanılarak YGYD'nün sürüm kararı gibi bir sonraki aşamasına geçiş kararlarına katkıda bulunur.

Test, yazılım geliştirme projesinde kullanıcıların dolaylı olarak temsil edilebilmesini sağlar. Test uzmanları, kullanıcı ihtiyaçlarının yazılım geliştirme yaşam döngüsü boyunca göz önünde bulundurulduğundan emin olurlar. Bunun alternatifi, yazılım geliştirme projesinin bir parçası olarak temsili bir kullanıcı grubunu sürece dahil etmek olsa da yüksek maliyetler ve uygun kullanıcıların bulunamaması nedeniyle bu genellikle mümkün değildir.

Yazılım testleri aynı zamanda sözleşmeye bağlı veya yasal gereksinimleri ya da düzenleyici standartları karşılamak için de gerekli olabilir.

1.2.2. Test Etme ve Kalite Güvence (KG)

Her ne kadar "test etme" ve "kalite güvence" (KG) genellikle birbirinin yerine kullanılan terimler olsa da test etme ve kalite güvence aynı şey değildir. Test etme süreci bir çeşit kalite kontrolüdür (KK).

Kalite kontrol, uygun kalite seviyelerine ulaşmayı destekleyen aktivitelere odaklanan ürün odaklı, düzeltici bir yaklaşımdır. Test, kalite kontrolün önemli bir ayağıdır, diğer bileşenler ise resmi yöntemleri (model kontrolü ve doğruluk kanıtı), simülasyonu ve prototiplemeyi içerir.

KG, süreçlerin uygulanması ve iyileştirilmesine odaklanan süreç odaklı ve önleyici bir yaklaşımdır. İyi bir sürecin doğru uygulanıp uygulanmadığı esasına göre çalışır ve sonuçta iyi bir ürün ortaya çıkmasına odaklanır. KG hem yazılım geliştirme hem de test süreci için geçerli olup projedeki herkesin sorumluluğundadır.

Test sonuçları KG ve KK sürecinde kullanılır. KK sürecinde test sonuçları hataların düzeltilmesinde kullanılır; KG sürecinde ise yazılım geliştirme ve test süreçlerinin ne kadar iyi gerçekleştirildiği hakkında geri bildirim sağlar.

1.2.3. İnsan Hataları, Hatalar, Arızalar ve Kök Nedenler

İnsanlar hata (yanlışlık) yapar ve bu da yazılım hatalarına (arıza, hata) neden olur ve sonunda başarısızlık meydana gelir. İnsanlar zaman baskısı, işlerin karmaşıklığı, süreçler, altyapı veya iletişim gibi birçok sebeple veya sadece yorgun oldukları ya da yeterli eğitime sahip olmadıkları için hata yapabilir.

Hatalar, gereksinim veya test betiği gibi dokümantasyonda, kaynak kodda veya derleme dosyası gibi destekleyici bir eserde bulunabilir. YGYD'nün ilk aşamalarında üretilen ürünlerdeki hatalar tespit edilmezse genellikle yaşam döngüsünün ilerleyen aşamalarında da hatalı ürünlerin ortaya çıkmasına sebep olur. Koddaki bir hata çalıştırılırsa sistem yapması gerekeni yapmayabilir veya yapmaması gereken bir şeyi yaparak bir arızaya neden olabilir. Bazı hatalar sistem çalıştırıldığında her zaman arızaya neden olurken, bazıları belirli koşullar altında arızaya neden olur, bazıları ise hiçbir zaman arızaya neden olmayabilir.

Arızaların tek sebebi insan hataları ve yazılım hataları değildir. Arızalar, radyasyon veya elektromanyetik alanın donanım yazılımında hatalara neden olması gibi çevresel koşullardan da kaynaklanabilir.

Kök neden, bir problemin ortaya çıkmasının temel sebebidir (ör. hataya yol açan bir durum). Kök nedenler genellikle bir arıza olduğunda veya bir hata tespit edildiğinde gerçekleştirilen kök neden analiziyle belirlenir. Kök nedenin ortadan kaldırılmasıyla benzer arızalar veya hatalar önlenebilir veya meydana gelme sıklıkları azaltılabilir.

1.3. Test Prensipleri

Yıllar içinde tüm yazılım testleri için geçerli genel bir rehber sunan bir dizi test prensibi ortaya atılmıştır. Bu ders programında yedi prensip açıklanmaktadır.

1. Testin amacı, yazılımda hataların olduğunu göstermektir; yazılımda hata kalmadığını ispatlamak değildir. Testler, test nesnesinde hataların mevcut olduğunu gösterebilir ancak hiç hata kalmadığını ispatlayamaz (Buxton 1970). Testler, test nesnesinde keşfedilmemiş hataların kalma olasılığını azaltır. Yazılımda yeni hatalar bulunamasa bile bu durum test nesnesinin, kullanıcıların ihtiyaçlarını karşılayacağı anlamına gelmez.

2. Yazılımı %100 test etmek imkansızdır Yazılımdaki her unsuru test etmek çok basit yazılımlar dışında imkansızdır (Manna 1978). Herşeyi test etmeye çalışmak yerine, testten daha çok verim almak için test teknikleri (bkz. konu 4), test senaryosu önceliklendirme (bkz. bölüm 5.1.5) ve risk bazlı test yaklaşımı (bkz. bölüm 5.2) kullanılmalıdır.

3. Erken test, zaman ve para tasarrufu sağlar. Hataların sürecin erken aşamalarında giderilmesi, yazılımda sonradan hata ortaya olasılığını düşürür. Bu da YGYD'nün sonraki aşamalarında daha az arızaya sebep olacağı için kalite maliyetini azaltacaktır (Boehm 1981). Hataları erken bulmak için hem statik test (bkz. konu 3) hem de dinamik test (bkz. konu 4) mümkün olduğunca erken başlatılmalıdır.

4. Hatalar yazılımın belli alanlarında yoğunlaşır. Bulunan hataların büyük çoğunluğu genellikle çok az sayıda sistem bileşeninde bulunur veya çok az sayıda sistem bileşeni canlı kullanım sırasında ortaya çıkan arızalara sebep olur. (Enders 1975). Bu olgu

Pareto prensibinin bir göstergesidir. Öngörülen hata kümeleri ile test ve canlı kullanım sırasında gözlemlenen gerçek hata kümeleri, risk-bazlı test için önemli bir girdidir (bkz. bölüm 5.2).

5. Antibiyotik direnci. Aynı testler birçok kez tekrarlandığında yeni hataları bulmada giderek daha etkisiz olurlar (Beizer 1990). Bu etkiyi önlemek için mevcut testler ve test verilerinin değiştirilmesi ve yeni testlerin yazılması gerekebilir. Ancak bazı durumlarda aynı testleri tekrar etmek faydalı olabilir (ör. otomatik regresyon testleri) (bkz. bölüm 2.2.3).

6. Yazılım testi, projenin bağlamına, koşullarına göre değişiklik gösterir. Test konusunda evrensel olarak uygulanan tek bir yaklaşım yoktur. Test, farklı proje bağlamlarında farklı şekillerde gerçekleştirilir (Kaner 2011).

7. Yeni hata bulamıyoruz başarılı bir yazılım elde ettik yanılgısı. Testin bir yazılımın başarısını garantileyeceğini beklemek bir yanılgıdır (yanlış bir kanıdır). Belirlenen tüm gereksinimleri titizlikle test etmek ve bulunan tüm hataları çözmek, kullanıcıların ihtiyaçlarını ve beklentilerini karşılamayan, müşterinin iş hedeflerine ulaşmasına yardımcı olmayan veya diğer rakip yazılımlara kıyasla daha zayıf bir yazılımın üretilmesini engelleyemeyebilir. Test sırasında doğrulamanın yanında sağlama da yapılmalıdır (Boehm 1981).

1.4. Test Aktiviteleri, Test Yazılımı ve Test Rolleri

Test süreci bağlama bağlıdır ancak daha kapsamlı ele alınacak olursa testlerin test hedeflerine ulaşma olasılığını artıran genel test aktivitesi grupları vardır. Bu test aktivitesi grupları bir test sürecini oluşturur. Test süreci çeşitli faktörlere dayalı olarak belirli bir duruma göre özelleştirilebilir. Hangi test sürecine hangi test aktivitelerinin dahil edileceğine, bunların nasıl uygulanacağına ve ne zaman gerçekleştirileceğine hazırlanan test planlaması kapsamında karar verilir (bkz. bölüm 5.1).

Aşağıdaki bölümlerde test aktiviteleri ve görevler, bağlamın etkisi, test yazılımı, test esası ile test yazılımı arasındaki izlenebilirlik ve test rolleri açısından bu test sürecinin genel yönleri açıklanmaktadır.

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

1.4.1. Test Aktiviteleri ve Görevleri

Bir test süreci genellikle aşağıdaki ana aktivite gruplarından oluşur. Her ne kadar bu aktivitelerin çoğu mantıklı bir sıra izliyor gibi görünse de bunlar genellikle tekrarlanarak veya paralel olarak uygulanır. Bu test aktivitelerinin genellikle sistem ve projeye uyarlanması gerekir.

Test planlama test hedefini belirleyip ardından genel bağlam dahilinde gerekli olan kısıtlar kapsamında hedeflere en iyi şekilde ulaşılmasını sağlayacak yaklaşımı seçmektir. Test planlama bölüm 5.1'de ayrıntılı bir şekilde açıklanmıştır.

Test gözetimi ve kontrolü. Test gözetimi tüm test aktivitelerinin devamlı izlenmesini ve planlanan ile gerçekleşenin karşılaştırılmasını içerir. Test kontrol test hedeflerine ulaşmak için gerekli aksiyonları almayı içerir. Test gözetimi ve kontrolü bölüm 5.3'te ayrıntılı bir şekilde açıklanmıştır.

Test analizi test edilebilir özellikleri belirlemek ve ilgili riskler ve risk seviyeleri ile birlikte ilişkili test koşullarını tanımlamak ve önceliklendirmek üzere test esasının analiz edilmesini içerir (bkz. bölüm 5.2). Test esası ve test hedefleri aynı zamanda içerebilecekleri hataları belirlemek ve test edilebilirliği analiz etmek için de değerlendirilir. Test analizi genellikle test tekniklerinin kullanımıyla desteklenir (bkz. konu 4). Test analizi ölçülebilir kapsama kriterleri açısından "ne test edilecek?" sorusuna cevap verir.

Test tasarımı test koşullarının test senaryoları ve diğer test yazılımları halinde detaylandırılmasını içerir

(ör. test başlatma belgeleri). Bu aktivite genellikle test senaryosu girdilerini belirlemeye yönelik bir rehber işlevi gören kapsam öğelerinin belirlenmesini içerir. Test teknikleri (bkz. konu 4) bu aktiviteyi desteklemek için kullanılabilir. Test tasarım aynı zamanda test verisi gereksinimlerinin belirlenmesi, test ortamının tasarlanması ve diğer gerekli altyapı ve araçların tespitini içerir. Test tasarımı "nasıl test edilecek" sorusunu cevaplar.

Test uyarlama test koşumu için gerekli test yazılımını oluşturmayı veya edinmeyi içerir (bkz. test verisi). Test senaryoları test prosedürleri şeklinde düzenlenebilir ve test grupları halinde birleştirilebilir. Manuel ve otomatik test betikleri oluşturulur. Test prosedürleri, etkin test koşumu için test koşum çizelgesi kapsamında önceliklendirilip düzenlenir (bkz. bölüm 5.1.5). Test ortamı oluşturulur ve kurulumunun uygun bir şekilde yapıldığı doğrulanır.

Test koşumu testleri test koşum çizelgesine göre çalıştırmayı içerir. Test koşumu manuel veya otomatik olabilir. Test koşumu sürekli test veya eşli test oturumları gibi birçok şekilde gerçekleşebilir. Gerçek test sonuçları beklenen sonuçlarla karşılaştırılır. Test sonuçları kaydedilir. Olası nedenlerin belirlenmesi için anomaliler analiz edilir. Bu analiz gözlemlenen arızlara dayalı olarak anomalileri raporlamamızı sağlar (bkz. bölüm 5.5).

Test tamamlama aktiviteleri genellikle çözülmemiş hatalar, değişiklik talepleri veya oluşturulan ürün iş listesi öğelerine ilişkin olarak proje kilometre taşlarında (ör. sürüm, döngü sonu, test seviyesi tamamlama) gerçekleşir. Gelecekte faydalı olabilecek test yazılımları belirlenir ve arşivlenir ya da uygun ekiplere devredilir. Test ortamı kararlaştırıldığı şekilde kapatılır. Test aktiviteleri tecrübeleri ve gelecekteki döngüler, sürümler veya projelere yönelik iyileştirmeleri belirlemek için analiz edilir (bkz. bölüm 2.1.6). Test tamamlama raporu oluşturulur ve paydaşlara iletilir.

1.4.2. Proje Bağlamında Test Süreci

Test izole şekilde yapılmaz. Test aktiviteleri organizasyon içinde gerçekleştirilen yazılım geliştirme süreçlerinin ayrılmaz bir parçasıdır. Testler de proje paydaşları tarafından finanse edilir ve nihai hedefi proje paydaşlarının iş ihtiyaçlarını karşılamaya yardımcı olmaktır. Dolayısıyla testlerin yapılma şekli aşağıdakiler de dahil birçok bağlamsal faktöre bağlı olacaktır:

- Paydaşlar (ihtiyaçlar, beklentiler, gereksinimler, iş birliği isteği vb.)
- Ekip üyeleri (beceri, bilgi, deneyim seviyesi, elverişlilik, eğitim ihtiyaçları vb.)
- Kurumun faaliyet alanı (test nesnesinin kritiklik düzeyi, tanımlanan riskler, pazar ihtiyaçları, yasal düzenlemeler vb.)
- Teknik faktörler (yazılım türü, ürün mimarisi, kullanılan teknoloji vb.)
- Proje kısıtları (kapsam, zaman, bütçe, kaynaklar vb.)
- Organizasyonel faktörler (organizasyonel yapı, mevcut politikalar, kullanılan pratikler vb.)
- Yazılım geliştirme yaşam döngüsü (mühendislik uygulamaları, geliştirme yöntemleri vb.)
- Araçlar (elverişlilik, kullanılabilirlik, uyumluluk vb.)

Bu faktörler test stratejisi, kullanılan test teknikleri, test otomasyonu derecesi, gerekli kapsam seviyesi, test dokümantasyonu ayrıntı düzeyi, raporlama vb. gibi testle ilgili konulara etki edecektir.

1.4.3. Test Yazılımı

Test yazılımı bölüm 1.4.1'de belirtilen test aktivitelerinden elde edilen çalışma ürünlerinin bir parçası olarak oluşturulur. Farklı organizasyonların çalışma ürünlerini nasıl ürettiği, şekillendirdiği, adlandırdığı, düzenlediği ve yönettiği konusunda

ciddi farklılılar vardır. Uygun yapılandırma yönetimi (bkz. bölüm 5.4) çalışma ürünleri arasında tutarlılık ve bütünlük sağlar. Aşağıda kapsamlı olmayan bir çalışma ürünleri listesi verilmiştir:

- **Test planlama çalışma ürünleri** şunları içerir: test planı, test zaman çizelgesi, risk kaydı ve giriş ve çıkış kriterleri (bkz. bölüm 5.1). Risk kaydı; risk olasılığı, risk etkisi ve riski azaltma bilgisini içeren bir riskler listesidir (bkz. bölüm 5.2). Test zaman çizelgesi, risk kaydı ile giriş ve çıkış kriterleri genellikle test planının bir parçasıdır.
- **Test gözetimi ve kontrolü çalışma ürünleri** şunları içerir: test ilerleme raporları (bkz. bölüm 5.3.2), kontrol direktifleri dokümantasyonu (bkz. bölüm 5.3) ve risk bilgisi (bkz. bölüm 5.2).
- **Test analizi çalışma ürünleri** şunları içerir: (önceliklendirilmiş) test koşulları (ör. kabul kriteri, bkz. bölüm 4.5.2) ve test esasındaki hatalara ilişkin hata raporu.
- **Test tasarımı çalışma ürünleri** şunları içerir: (önceliklendirilmiş) test senaryoları, test başlatma belgeleri, kapsam öğeleri, test verisi gereksinimleri ve test ortamı gereksinimleri.
- **Test uyarlama çalışma ürünleri** şunları içerir: test prosedürleri, otomatik test betikleri, test grupları, test verisi, test koşumu çizelgesi ve test ortamı öğeleri. Test ortamı öğelerine ilişkin örnekler: taklit uygulamalar, sürücüler, simülatörler ve servis sanallaştırması.
- **Test koşumu çalışma ürünleri** şunları içerir: test kayıtları ve hata raporları (bkz. bölüm 5.5).
- **Test tamamlama çalışma ürünleri** şunları içerir: test tamamlama raporu (bkz. bölüm 5.3.2), sonraki proje veya döngülerin iyileştirilmesine yönelik aksiyon öğeleri, tecrübeler ve değişiklik talepleri (ör. ürün iş listesi öğeleri).

1.4.4. Test Esası ve Test Yazılımı Arasında İzlenebilirlik

Etkili test gözetimi ve kontrolü uygulamak için test süreci boyunca test esası unsurları, bu unsurlarla ilişkili test yazılımı (ör. test koşulları, riskler, test senaryoları), test sonuçları ve tespit edilen hatalar arasında izlenebilirliği sağlamak ve sürdürmek önemlidir.

Doğru izlenebilirlik kapsam değerlendirmesini destekler ve dolayısıyla test esasında ölçülebilir kapsama kriterleri tanımlanmışsa bunların takip edilmesinde çok faydalı olur. Kapsama kriterleri test hedeflerine ne ölçüde ulaşıldığını gösteren aktiviteleri gerçekleştirmek açısından anahtar performans göstergeleri olarak işlev görebilir (bkz. bölüm 1.1.1). Örnek:

- Test senaryolarının gereksinimlere göre izlenebilirliğine bakarak test senaryolarının gereksinimleri karşıladığını doğrulamak mümkündür.
- Test sonuçlarının risklere göre izlenebilirliği bir test nesnesindeki kalan risk seviyesini değerlendirmede kullanılabilir.

İyi bir izlenebilirlik yalnızca kapsam değerlendirmesine değil, aynı zamanda etki analizine, test denetimlerine ve bilgi teknolojileri (BT) yönetişim gereksinimlerinin karşılanmasına da yardımcı olur. İyi bir izlenebilirlik, test esası unsurlarının durumunu da içererek test ilerleme durumu ve tamamlanma raporlarının anlaşılmasını kolaylaştırır. Aynı zamanda testin teknik yönlerinin paydaşlara anlaşılır şekilde bildirilmesine de yardımcı olur. İzlenebilirlik, iş hedeflerine göre ürün kalitesinin, süreç kapasitesinin ve proje ilerlemesinin değerlendirilmesi için bilgi sağlar

1.4.5. Test Etme Sürecindeki Roller

Bu ders programında testteki iki önemli role yer verilmiştir: test yönetimi rolü ve test etme rolü. Bu iki role sahip kişilere atanan faaliyetler ve görevler, projenin ve ürünün bağlamı gibi faktörlere, o rolü üstlenen kişilerin becerilerine ve kuruma göre değişir.

Test yönetimi rolü, genel olarak test sürecinden, test ekibinden ve test aktivitelerine liderlik edilmesinden sorumludur. Test yönetimi rolü temelde test planlama, test gözetimi ve kontrolü ve test tamamlama aktivitelerine odaklıdır. Test yönetimi rolünün yerine getirilme şekli, bağlama göre değişir. Örneğin, Çevik Yazılım Geliştirmede bazı test yönetimi görevleri Çevik ekip tarafından gerçekleştirilirken, birden fazla ekibi veya tüm organizasyonu ilgilendiren görevler geliştirme ekipleri dışındaki test yöneticileri tarafından yerine getirilebilir.

Test etme rolü testin mühendislik (teknik) yönünün tüm sorumluluğuna sahiptir. Test etme rolü temelde test analizi, test tasarımı, test uyarlama ve test koşumu aktivitelerine odaklanır.

Rollerin sorumluluğunu farklı zamanlarda farklı kişiler alabilirler. Örneğin, test yönetimi rolü bir ekip lideri, test yöneticisi, geliştirme yöneticisi vb. tarafından üstlenilebilir. Test etme ve test yönetimi rollerini aynı anda tek bir kişinin üstlenmesi de mümkündür.

1.5. Test Etme Sürecinde Gerekli Beceriler ve İyi Uygulamalar

Beceri; kişinin bilgi birikiminden, pratiğinden ve kabiliyetinden ileri gelen, bir işi iyi yapma yeteneğidir. İyi test uzmanları işlerini iyi yapmak için bazı gerekli becerilere sahip olmalıdır. İyi test uzmanları etkin birer ekip üyesi olmalı ve çeşitli test bağımsızlığı seviyelerinde testler gerçekleştirebilmelidir.

1.5.1. Test Etme Sürecinde Gerekli Genel Beceriler

Genel olmalarına rağmen, aşağıdaki beceriler test uzmanları için özellikle önemlidir:

- Test bilgisi (test etmenin etkinliğini artırmak için, ör. test teknikleri kullanarak)
- Titizlik, dikkat, merak, detaycı olmak, sistematik çalışmak (özellikle bulması zor olanlar başta olmak üzere hataları bulmak için)
- İyi iletişim becerileri, aktif dinleme, ekip oyuncusu olma (tüm paydaşlarla etkin iletişim kurmak, diğeriyle bilgi paylaşmak, anlaşılmak ve hataları raporlayıp görüşmek için)
- Analitik düşünce, eleştirel düşünce, yaratıcılık (test etmenin etkinliğini artırmak için)
- Teknik bilgi (test etmenin etkinliğini artırmak için, ör. uygun test araçlarını kullanarak)
- Alan bilgisi (son kullanıcıları/iş temsilcilerini anlayabilmek ve onlarla iletişim kurabilmek için)

Test uzmanları genellikle kötü haberi veren taraftadırlar. Kötü haber verenleri suçlamak ise tipik bir insan davranışıdır
Bu nedenle, test uzmanları için iletişim becerisi kritik önem taşır. Gerçekleştirilen testin sonucu, ürüne ve sahibine yöneltilmiş bir eleştiri olarak algılanabilir. Doğrulama yanlılığı, mevcut görüş ve inançlara uymayan bilgilerin kabul edilmesini zorlaştırabilir. Projenin başarısına ve ürün kalitesine büyük katkı sağlasa da, bazı insanlar testleri yıkıcı bir faaliyet olarak algılayabilir. Bu görüş doğrultusunda, hatalar ve arızalar hakkındaki bilgiler yapıcı bir şekilde iletilmelidir.

1.5.2. Tüm Ekip Yaklaşımı

Bir test uzmanı için en önemli becerilerden biri, bir ekip bağlamında etkin çalışma ve ekip hedeflerine olumlu katkı sağlama becerisidir. Ekstrem Programlama (bkz. bölüm 2.1) metodolojisinden elde edilen bir pratik olan tüm ekip yaklaşımı bu beceriyi temel alır.

Tüm ekip yaklaşımında, gerekli bilgi ve beceriye sahip tüm ekip üyeleri her türlü görevi yapabilir ve kaliteden herkes sorumludur. Ekip üyeleri aynı çalışma alanını (fiziksel veya sanal) paylaşır çünkü ortak çalışma alanı iletişim ve etkileşimi kolaylaştırır. Tüm ekip yaklaşımı ekip ilişkilerini geliştirir, ekip içinde iletişimi ve iş birliğini artırır ve ekibin farklı beceri setlerinin proje yararına kullanılmasına olanak sağlayarak sinerji yaratır.

Test uzmanları, istenen kalite seviyelerine ulaşılabilmesi için diğer ekip üyeleriyle yakın iş birliği içinde çalışırlar. Bu, uygun kabul testleri oluşturmalarına yardımcı olmak için iş birimleriyle iş birliği yapmanın yanı sıra test stratejisi ve test otomasyonu metodolojilerine karar vermek için yazılımcılarla birlikte çalışmayı da içerir. Böylece test uzmanları, test alanındaki bilgilerini diğer ekip üyelerine aktarabilir ve ürünün gelişimine katkı sağlayabilir.

Bağlama açısından tüm ekip yaklaşımı her zaman uygun olmayabilir. Örneğin, bazı emniyet hassasiyetli durumlarda, yüksek düzeyde test bağımsızlığı gerekli olabilir.

1.5.3. Testin Bağımsızlığı

Belli bir ölçüde bağımsızlık, yazarın ve test uzmanının bilişsel önyargıları arasındaki farklar nedeniyle test uzmanının hataları bulmada daha etkili olmasını sağlar (bkz. Salman 1995). Ancak bağımsızlık, aşina olmanın yerine geçemez. Örneğin yazılımcılar kendi kodlarındaki birçok hatayı verimli bir şekilde bulabilir.

Çalışma ürünleri; kendi yazarları (bağımsızlık yok), yazarın aynı ekipten arkadaşları (biraz bağımsız), yazarın ekibinin dışındaki fakat kurum içindeki test uzmanları (yüksek bağımsızlık) veya şirket dışından test uzmanları (çok yüksek bağımsızlık) tarafından test edilebilir. Çoğu proje için genellikle en iyi seçenek, testin birden çok bağımsızlık seviyesinde yapılmasıdır (ör. bileşen ve bileşen entegrasyon testi yazılımcılar tarafından, sistem ve sistem entegrasyon testi test ekibi tarafından, kabul testi ise iş temsilcileri tarafından yapılır).

Bağımsız testin başlıca faydası farklı geçmiş deneyimleri, teknik bakış açıları ve eğilimleri nedeniyle bağımsız test uzmanlarının, yazılımcılara kıyasla farklı çeşitteki arızaları ve hataları bulma olasılıklarının daha yüksek olmasıdır. Ayrıca bağımsız bir test uzmanı, sistemin analiz edilmesi ve hayata geçirilmesi sırasında paydaşların yaptığı varsayımları doğrulayabilir, sorgulayabilir veya çürütebilir.

Ancak bazı sakıncalar da söz konusudur. Bağımsız test uzmanları geliştirme ekibinden farklı hareket edebilir ve bu da iş birliği eksikliğine, iletişim problemlerine veya geliştirme ekibiyle olumsuz ilişkiler yaşanmasına yol açabilir. Yazılımcılar kalite konusunda sorumluluk bilincini kaybedebilir. Bağımsız test uzmanları bir dar boğaz olarak görülebilir veya sürümdeki gecikmelerden sorumlu tutulabilir.

2. Yazılım Geliştirme Yaşam Döngüsü Boyunca Test – 130 dakika

Anahtar kelimeler

kabul testi, kara kutu testi, bileşen entegrasyon testi, bileşen testi, onaylama testi, fonksiyonel test, entegrasyon testi, bakım testi, fonksiyonel olmayan test, regresyon testi, shift-left, sistem entegrasyon testi, sistem testi, test seviyesi, test nesnesi, test çeşidi, beyaz kutu testi

Konu 2 Öğrenme Hedefleri:

2.1 Yazılım Geliştirme Yaşam Döngüsü Bağlamında Test

FL-2.1.1 (K2) Seçilen yazılım geliştirme yaşam döngüsünün test üzerindeki etkisini açıklamak

FL-2.1.2 (K1) Tüm yazılım geliştirme yaşam döngüsü için geçerli olan iyi test uygulamalarını hatırlamak

FL-2.1.3 (K1) Geliştirmeye yönelik önce-test-et yaklaşımlarının örneklerini hatırlamak

FL-2.1.4 (K2) DevOps'un test üzerinde nasıl bir etkisi olabileceğini özetlemek

FL-2.1.5 (K2) Shift-left yaklaşımını açıklamak

FL-2.1.6 (K2) Geçmişe dönük verilerin süreç iyileştirmesi için nasıl bir mekanizma olarak kullanılabileceğini açıklamak

2.2 Test Seviyeleri ve Test Çeşitleri

FL-2.2.1 (K2) Farklı test seviyelerini ayırt etmek

FL-2.2.2 (K2) Farklı test çeşitlerini ayırt etmek

FL-2.2.3 (K2) Onaylama testleri ve regresyon testlerini birbirinden ayırmak

2.3 Bakım Testleri

FL-2.3.1 (K2) Bakım testleri ve tetikleyicilerini özetlemek

2.1. Yazılım Geliştirme Yaşam Döngüsü Bağlamında Test

Yazılım geliştirme yaşam döngüsü (YGYD) yazılım geliştirme sürecinin kavramsal, üst düzey bir tasviridir. Bir YGYD modeli, bu süreçte gerçekleştirilen farklı geliştirme aşamaları ve faaliyet türlerinin hem mantıksal hem de kronolojik olarak birbirleriyle nasıl ilişkili olduğunu gösterir. YGYD modellerine örnekler: sıralı yazılım geliştirme modelleri (ör. şelale modeli, V modeli), döngüsel geliştirme modeli (ör. spiral model, prototipleme) ve artımlı geliştirme modelleri (ör. Birleşik Süreç).

Yazılım geliştirme süreçlerindeki bazı aktiviteler daha ayrıntılı yazılım geliştirme yöntemleri ve Çevik uygulamalarla da tanımlanabilir. Örnekler: kabul testi güdümlü yazılım geliştirme (ATDD), davranış güdümlü yazılım geliştirme (BDD), alan güdümlü tasarım (DDD), ekstrem programlama (XP), özellik güdümlü geliştirme (FDD), Kanban, Yalın BT, Scrum ve test güdümlü geliştirme (TDD).

2.1.1. Yazılım Geliştirme Yaşam Döngüsünün Test Üzerindeki Etkisi

Başarılı olması için testlerin YGYD'ne uyarlanması gerekir. YGYD seçimi aşağıdakilere etki eder:

- Test aktivitelerinin kapsamı ve zamanlaması (ör. test seviyeleri ve test çeşitleri)
- Test dokümantasyonun ayrıntı düzeyi
- Test tekniklerinin ve test yaklaşımının seçimi
- Test otomasyonunun kapsamı
- Test uzmanının rol ve sorumlulukları

Sıralı yazılım geliştirme modellerinde, ilk aşamalarda test uzmanları genellikle gereksinim gözden geçirmelerine, test analizine ve test tasarımına katılır. Çalıştırılabilir kod genellikle sonraki aşamalarda oluşturulur, bu nedenle tipik olarak dinamik testler YGYD'nün erken aşamalarında gerçekleştirilemez.

Bazı döngüsel ve artımlı geliştirme modellerinde her döngünün bir çalışma prototipi veya ürün özelliği sağladığı varsayılır. Bu da her döngüde tüm test seviyelerinde hem statik hem de dinamik testlerin gerçekleştirilebileceği anlamına gelir. Yazılım özelliklerinin sık hayata geçirilmesi hızlı geri bildirim ve kapsamlı regresyon testleri gerektirir.
Çevik yazılım geliştirme, projenin süresi boyunca değişikliklerin olabileceğini varsayar. Bu nedenle, çevik projelerde iş ürünü belgelerinin hafifletilmesi ve regresyon testlerini kolaylaştırmak için kapsamlı test otomasyonu tercih edilir. Ayrıca, manuel testlerin çoğu, tecrübeye dayalı test teknikleri kullanılarak yapılır ve bu teknikler, önceden kapsamlı test analizi ve tasarım gerektirmez (bkz. Bölüm 4.4).

2.1.2. Yazılım Geliştirme Yaşam Döngüsü ve İyi Test Etme Uygulamaları

Seçilen YGYD modelinden bağımsız iyi test uygulamaları aşağıdakileri içerir:

- Her yazılım geliştirme aktivitesine karşılık gelen bir test aktivitesi vardır ve böylece tüm geliştirme aktivitelerinin kalite kontrole tabi olması sağlanır
- Farklı test seviyeleri (bkz. konu 2.2.1) belirli ve farklı test hedeflerine sahiptir ve bu da testin uygun şekilde kapsamlı olmasına olanak tanırken gereksiz tekrarlardan kaçınmayı sağlar
- Belirli bir test seviyesi için test analizi ve tasarımı YGYD'nün ilgili geliştirme aşamasında başlar ve böylece test "erken test" prensibine uygun şekilde test gerçekleşebilir. (bkz. bölüm 1.3)

- Test uzmanları, dokümantasyonun taslakları hazır olur olmaz çalışma ürünlerini gözden geçirme sürecine dahil olur, böylece erken test ve hata tespiti “shift-left” stratejisini destekleyebilir (bkz. bölüm 2.1.5).

2.1.3. Yazılım Geliştirme Faktörü Olarak Test

TDD, ATDD ve BDD benzer yazılım geliştirme yaklaşımlarıdır, burada testler geliştirme sürecini yönlendiren bir araç olarak tanımlanır. Bu yaklaşımların her biri erken test etme prensibini uygular (bkz. bölüm 1.3) ve shift-left yaklaşımını izler (bkz. bölüm 2.1.5), çünkü testler kod yazılmadan önce tanımlanır. Döngüsel bir geliştirme modelini desteklerler. Bu yaklaşımların karakteristik özellikleri aşağıdaki gibidir:

Test Güdümlü Yazılım Geliştirme (TDD):

- Kodlamayı test senaryoları aracılığıyla yönlendirir (kapsamlı yazılım tasarımı yerine) (Beck 2003)
- Önce testler yazılır, sonra testleri yerine getirmek için kod yazılır ve ardından testler ve kod yeniden düzenlenir

Kabul Testi Güdümlü Yazılım Geliştirme (ATDD) (bkz. bölüm 4.5.3):
- Kabul kriterlerinden testler türetilir ve bu, sistem tasarım sürecinin bir parçası olarak yapılır (Gärtner 2011) - Testler, uygulamanın ilgili bölümü geliştirilmeden önce yazılır ve ardından testleri karşılayacak şekilde kodlama yapılır.

Davranış Güdümlü Yazılım Geliştirme (BDD):
- Uygulamanın istenen davranışını, genellikle Given/When/Then formatı kullanılarak, paydaşlar tarafından anlaşılması kolay bir doğal dilin basit bir formunda yazılan test senaryoları ile ifade eder. (Chelimsky 2010)
- Test senaryoları daha sonra otomatik olarak yürütülebilir testlere çevrilir

Yukarıdaki yaklaşımların tümünde, testler gelecekteki uyarlama/yeniden düzenleme süreçlerinde kod kalitesini sağlamak için otomatikleştirilmiş testler olarak kalabilir.

2.1.4. DevOps ve Test Etme

DevOps, ortak hedeflere ulaşmak için yazılım geliştirme (test dahil) ve operasyonun birlikte çalışmasını sağlayarak sinerji oluşturmayı amaçlayan organizasyonel bir yaklaşımdır. DevOps, yazılım geliştirme (test dahil) ve operasyon arasında köprü görevi görmek ve işlevlerine eşit değerde muamele etmek için kuruluş içinde kültürel bir değişim gerektirir. DevOps ekip özerkliğini, hızlı geri bildirimi, entegre araç zincirlerini ve sürekli entegrasyon (CI) ve sürekli teslimat (CD) gibi teknik uygulamaları destekler. Bu, ekibin DevOps teslimat hattı aracılığıyla yüksek kaliteli kodları daha hızlı oluşturmasını, test etmesini ve yayınlamasını sağlar (Kim 2016).

Test perspektifinden bakıldığında DevOps'un bazı faydaları şunlardır:

- Kod kalitesi hakkında hızlı geri bildirim sağlar ve değişikliklerin mevcut kodu olumsuz etkileyip etkilemediğini belirler.
- Sürekli teslimat (CI), yazılım geliştiricileri bileşen testleri ve statik analiz eşliğinde yüksek kaliteli kodlar sunmaya teşvik ederek test konusunda shift-left yaklaşımı uygulanmasını destekler (bkz. bölüm 2.1.5).
- Stabil test ortamları oluşturmayı kolaylaştıran CI/CD gibi otomatik süreçleri destekler
- Fonksiyonel olmayan gereksinimlere (ör. performans ve güvenilirlik) ilişkin görüş alanını artırır
- Teslimat hattında yapılan test otomasyonu, tekrarlayan manuel testlere ihtiyacı azaltır

- Otomatik regresyon testlerinin ölçeği ve kapsamı dolayısıyla regresyon riski en aza indirilir

DevOps'un risk ve zorlukları da vardır; bunlar arasında:

- DevOps teslimat hattı tanımlanmalı ve kurulmalıdır
- CI / CD araçları tanıtılmalı ve bakımı yapılmalıdır
- Test otomasyonu ek kaynaklar gerektirir. Test otomasyonunun kurması ve sürdürmesi zor olabilir

Her ne kadar DevOps kapsamında yüksek düzeyde test otomasyonu bulunsa da özellikle kullanıcı perspektifinden bakıldığında manuel testlere yine de ihtiyaç olacaktır.

2.1.5. Shift-Left Yaklaşımı

Testin YGYD'nün başlarında gerçekleştirildiği bir yaklaşım olan “erken test” prensibi (bkz. bölüm 1.3) bazen shift-left olarak adlandırılır. Shift-left genellikle testlerin daha erken yapılmasını önerir (örneğin, kodun yazılmasını veya bileşenlerin entegre edilmesini beklemeden), ancak bu YGYD’nün ilerleyen aşamalarında testlerin ihmal edilmesi gerektiği anlamına gelmez.

Testlerde "shift-left" yaklaşımının nasıl sağlanacağını gösteren birtakım iyi uygulamalar vardır:

- Test bakış açısıyla analizin gözden geçirilmesi. Analize ilişkin bu gözden geçirme aktiviteleri genelde belirsizlikler, eksiklikler ve tutarsızlıklar gibi potansiyel hataları tespit eder.
- Kod yazılmadan önce test senaryolarının yazılması ve kodun yazılması sırasında kodun bir test kuluçkasında çalıştırılması
- Kod deposuna gönderildiğinde kaynak koda eşlik edecek hızlı geri bildirim ve otomatik bileşen testleri sağladığı için CI'nin ve hatta daha da iyisi CD'nin kullanılması
- Dinamik test öncesinde veya otomatik süreç kapsamında kaynak kodun statik analizinin tamamlanması
- Mümkün olan durumlarda bileşen testi seviyesinden başlayarak fonksiyonel olmayan testlerin gerçekleştirilmesi. Bu bir tür shift-left uygulamasıdır. Çünkü fonksiyonel olmayan test türleri, YGYD’nün sonraki aşamalarında, eksiksiz bir sistem ve temsili bir test ortamı mevcut olduğunda gerçekleştirilmektedir.

Bir shift-left yaklaşımı süreç başında ilave eğitim, efor ve/veya maliyete yol açabilir ancak bu yaklaşımın sürecin sonraki aşamalarında efor ve/veya maliyet tasarrufu sağlaması beklenir.

Shift-left yaklaşımı açısından paydaşların ikna edilmesi ve bu konsepte inanmaları önemlidir.

2.1.6. Geçmişe Dönük Öğeler ve Süreç İyileştirmesi

Geçmişe dönük öğeler ("proje sonrası toplantı" ve geriye dönük proje kazanımları olarak da bilinir) genelde bir proje veya döngü sonunda, bir sürüm kilometre taşında veya gerektiğinde ele alınır. Geçmişe dönük öğelerin zamanlaması ve organizasyonu, izlenen YGYD modeline bağlıdır. Bu toplantılarda katılımcılar (sadece test uzmanları değil, aynı zamanda ör. geliştiriciler, mimarlar, ürün sahibi ve iş analistleri) şunları görüşür:

- Başarılı olan neydi ve bu nasıl sürdürülmeli?
- Başarısız olan neydi ve bu konuda nasıl bir iyileştirme yapılabilir?

- Başarıları nasıl sürdürebilir ve ileriye dönük iyileştirmeleri nasıl uygulayabiliriz?

Sonuçlar kaydedilmelidir ve normalde test tamamlama raporunun bir parçasıdır (bkz. bölüm 5.3.2). Geçmişe dönük öğeler sürekli iyileştirmenin başarıyla uygulanmasında kritiktir ve tavsiye edilen her tür iyileştirmenin takip edilmesi önemlidir.

Test sürecine tipik faydaları şunlardır:

- Daha yüksek test etkinliği / verimliliği (ör. süreç iyileştirmesi önerilerinin uygulanması)
- Daha yüksek test yazılımı kalitesi (ör. test süreçlerinin birlikte gözden geçirilmesi)
- Ekip kaynaşması ve öğrenmesi (ör. sorun bildirme ve iyileştirme noktaları önerme fırsatının bir sonucu olarak)
- Daha yüksek test esası kalitesi (ör. gereksinimlerin kapsam ve kalitesindeki eksikliklerin ele alınıp çözülmesi)
- Geliştirme ve test etme süreci arasında daha iyi iş birliği (ör. iş birliğinin düzenli olarak incelenip optimize edilmesi)

2.2. Test Seviyeleri ve Test Çeşitleri

Test seviyeleri, birlikte düzenlenen ve yönetilen test aktivitesi gruplarıdır. Her test seviyesi, belirli bir yazılım geliştirme aşamasındaki yazılımla ilgili olarak gerçekleştirilen, bağımsız bileşenlerden komple sistemlere veya bazı durumlarda sistemlerin sistemlerine kadar gerçekleştirilen test sürecinin bir örneğidir.

Test seviyeleri, YGYD içindeki diğer aktivitelerle bağlantılıdır. Sıralı YGYD modellerinde, test seviyeleri genellikle bir seviyenin çıkış kriterleri bir sonraki seviyenin giriş kriterlerinin bir parçası olacak şekilde tanımlanır. Bazı döngüsel modellerde bu geçerli olmayabilir. Geliştirme aktiviteleri birden fazla test seviyesini kapsayabilir. Test seviyeleri zaman içinde çakışabilir.

Test çeşitleri gereksinimlerle ilgili test aktivitesi gruplarıdır ve bu test aktivitelerinin çoğu her test seviyesinde yapılabilir.

2.2.1. Test Seviyeleri

Bu ders programında aşağıdaki beş test seviyesi açıklanmaktadır:

- **Bileşen testi** (birim testi olarak da bilinir), bileşenleri ayrı olarak test etmeye odaklanır. Genelde test kuluçkası veya birim testi çerçeveleri gibi özel destek gerektirir. Bileşen testi normalde geliştiriciler tarafından kendi geliştirme ortamlarında yapılır.
- **Bileşen entegrasyon testi** (birim entegrasyon testi olarak da bilinir) arayüzlerin ve bileşenler arasındaki etkileşimlerin testine odaklanır. Bileşen entegrasyon testi yoğunlukla aşağıdan yukarıya, yukarıdan aşağıya veya big-bang gibi entegrasyon stratejisi yaklaşımlarına bağlıdır.
- **Sistem testi** bir sistemin veya ürünün genel davranışına ve yeteneklerine odaklanır, genellikle uçtan uca görevlerin fonksiyonel ve fonksiyonel olmayan testlerini içerir. Bazı fonksiyonel olmayan gereksinimler için, bunların temsili bir test ortamında komple bir sistem üzerinde test edilmesi tercih edilir (ör. kullanılabilirlik). Alt sistemlerin

simülasyonları da kullanılabilir. Sistem testi bağımsız bir test ekibi tarafından yapılabilir ve sisteme yönelik gereksinimlerle ilgilidir.

- **Sistem entegrasyon testi** test edilen sistem ile diğer sistemler ve harici hizmetler arasındaki arayüzlerin test edilmesine odaklanır. Sistem entegrasyon testi için tercihen operasyonel ortama benzer uygun test ortamları gereklidir.
- **Kabul testi** sistemin sağlamasını yapmaya ve kullanıcıya sunulmaya hazır olduğunu göstermeye odaklanır. Bunun sonucunda sistemin kullanıcı ihtiyaçlarını yerine getirebilir durumda olduğu kanıtlanmış olur. İdeal kabul testi, hedeflenen kullanıcılar tarafından gerçekleştirilmelidir. Başlıca kabul testi çeşitleri şunlardır: kullanıcı kabul testi (KKT), operasyonel kabul testi, sözleşmeye dayalı ve yasal düzenlemeye dayalı kabul testi, alfa testi ve beta testi.

Test seviyelerini test aktivitelerinden ayırmak için aşağıdaki liste kullanılabilir:

- Test nesnesi
- Test hedefleri
- Test esası
- Hatalar ve arızalar
- Yaklaşım ve sorumluluklar

2.2.2. Test Çeşitleri

Birçok test çeşidi mevcuttur ve projelerde uygulanabilir. Bu ders programında aşağıdaki dört test çeşidi ele alınmıştır:

Fonksiyonel testler bir bileşen ya da sistemin yerine getirmesi gereken fonksiyonları değerlendirir. Fonksiyonlar, test nesnesinin "ne" yapması gerektiğini tanımlar. Fonksiyonel testin ana hedefi fonksiyonel bütünlüğü, fonksiyonel doğruluğu ve fonksiyonel uygunluğu kontrol etmektir.

Fonksiyonel olmayan testler bir bileşen veya sistemin fonksiyonel gereksinimleri dışındaki özelliklerini değerlendirir. Fonksiyonel olmayan testler sistemin yapılması gerekenleri "ne kadar iyi" yaptığını ölçümlemeye çalışır. Fonksiyonel olmayan testin ana hedefi fonksiyonel olmayan gereksinimleri kontrol etmektir. ISO/IEC 25010 standardı, fonksiyonel olmayan gereksinimlere ilişkin aşağıdaki sınıflandırmayı sunar:

- Performans
- Uyumluluk
- Kullanılabilirlik
- Güvenilirlik
- Güvenlik
- Sürdürülebilirlik
- Taşınabilirlik

Fonksiyonel olmayan testlerin yaşam döngüsünün erken aşamalarında başlaması bazen uygun olabilir (ör. gözden geçirmelerin ve bileşen testlerinin veya sistem testlerinin bir parçası olarak). Birçok fonksiyonel olmayan test, fonksiyonel testlerden türetilmiştir.

Aynı fonksiyonel testleri kullanırlar, ancak fonksiyonu yerine getirirken fonksiyonel olmayan bir kısıtlamanın karşılanıp karşılanmadığını kontrol ederler (ör. bir fonksiyonun belirli bir süre içinde çalışıp çalışmadığını veya bir fonksiyonun yeni bir platforma taşınıp taşınamayacağını kontrol etmek gibi). Fonksiyonel olmayan hataların geç fark edilmesi bir projenin başarısı için ciddi tehdit oluşturabilir. Fonksiyonel olmayan testler bazen kullanılabilirlik testi için kullanılabilirlik laboratuvarı gibi çok özel bir test ortamına ihtiyaç duyar.

Kara kutu testi (bkz. bölüm 4.2) gereksinim bazlıdır ve test nesnesine dışarıdan bakarak testleri elde eder. Kara kutu testinin ana hedefi, sistemin gereksinimlere karşı davranışını kontrol etmektir.

Beyaz kutu testi (bkz. bölüm 4.3) yapı bazlıdır ve sistemin uygulama veya iç yapısından testler elde eder (ör. kod, mimari, iş akışları ve veri akışları). Beyaz kutu testinin ana hedefi, testlerin altta yatan yapıyı kabul edilebilir seviyeye kadar kapsamasını sağlamaktır.

Yukarıda belirtilen dört test çeşidinin hepsi, her seviyede odak noktası farklı olsa da, tüm test seviyelerine uygulanabilir. Bahsedilen test çeşitlerinin her biri için, test koşulları ve test senaryoları elde etmek üzere çeşitli test teknikleri uygulanabilir.

2.2.3. Onaylama Testleri ve Regresyon Testleri

Değişiklikler genelde yeni bir özellik ekleyerek bileşeni veya sistemi geliştirmek ya da bir hatayı gidererek bileşeni veya sistemi düzeltmek için yapılır. Test etme süreci ayrıca onaylama testi ve regresyon testini de içermelidir.

Onaylama testi asıl hatanın başarıyla giderildiğini onaylamak için yapılır. Riske bağlı olarak, yazılımın düzeltilmiş versiyonu aşağıdakiler de dahil olmak üzere çeşitli şekillerde test edilebilir:

- daha önce hata nedeniyle başarısız olmuş tüm test senaryolarını çalıştırarak veya
- hatayı düzeltmek için gereken değişiklikleri kapsayacak şekilde yeni testler ekleyerek

Ancak, hataları düzeltirken zaman veya para kısıtlı olduğunda onaylama testi, hatanın neden olduğu arızayı yeniden üretecek adımları uygulamak ve hatanın meydana gelmediğini kontrol etmekle sınırlı olabilir.

Regresyon testi halihazırda onaylama testinden geçirilmiş bir düzeltme de dahil olmak üzere bir değişikliğin herhangi bir olumsuz sonuca neden olmadığını teyit eder. Bu olumsuz sonuçlar, değişikliğin yapıldığı bileşeni, aynı sistemdeki diğer bileşenleri ve hatta diğer bağlı sistemleri etkileyebilir. Regresyon testi, test nesnesinin kendisiyle olduğu kadar ortamla da ilgili olabilir. Regresyon testinin kapsamını en iyi duruma getirmek için öncelikle bir etki analizi yapılması tavsiye edilir. Etki analizi, yazılımın hangi kısımlarının etkilenebileceğini gösterir.

Regresyon test grupları birçok kez çalıştırılır ve genellikle regresyon test senaryolarının sayısı her döngüde veya sürümde artar, bu nedenle regresyon testi test otomasyonu için güçlü bir adaydır. Bu testlerin otomasyonu projenin erken aşamalarında başlamalıdır. DevOps (bkz. bölüm 2.1.4) gibi CI kullanılan durumlarda, otomatik regresyon testlerini de dahil etmek iyi bir uygulamadır. Duruma bağlı olarak bu, farklı seviyelerde regresyon testleri içerebilir.

Test seviyelerinde hatalar düzeltilirse ve/veya değişiklikler yapılırsa tüm test seviyelerinde test nesnesi için onaylama testi ve/veya regresyon testi gereklidir.

2.3. Bakım Testleri

Farklı bakım kategorileri vardır. Bakım düzeltici olabilir, ortamdaki değişikliklere uyarlanabilir veya performansı veya sürdürülebilirliği artırabilir (ayrıntılar için bkz. ISO/IEC 14764). Bu nedenle bakım, planlı sürümleri/dağıtımları ve planlanmamış sürümleri/dağıtımları (düzeltmeler) içerebilir. Bir değişiklik yapılmadan önce, sistemin diğer alanlarında yaratacağı potansiyel sonuçlara dayanarak değişikliğin yapılıp yapılmayacağına karar vermek için etki analizi yapılabilir. Canlı ortamdaki bir sistemde yapılan değişikliklerin test edilmesi, hem değişikliğin uygulanma başarısının değerlendirilmesini hem de sistemin değişmeyen kısımlarında (ki bu genellikle sistemin büyük bir kısmıdır) olası regresyonların kontrol edilmesini içerir.

Bakım testinin kapsamı genellikle şunlara bağlıdır:

- Değişikliğin risk derecesi
- Mevcut sistemin boyutu
- Değişikliğin boyutu

Bakım ve bakım testleri tetikleyicileri aşağıdaki gibi sınıflandırılabilir:

- Değişiklikler: planlı iyileştirmeler (ör. sürüm bazlı), düzeltici değişiklikler veya düzeltmeler.
- Operasyonel ortam yükseltmeleri veya taşımaları: yeni ortam ve değiştirilen yazılımla ilgili testler veya başka bir uygulamadan gelen verilerin bakımı yapılan sisteme geçirilmesine yönelik testler gerektirebilen, bir platformdan diğerine taşıma gibi durumlar.
- Kullanımdan kaldırma: bir uygulamanın kullanım ömrünün sonuna gelinmesi gibi durumlar. Bir sistemin kullanımı sonlandırılırken, verilerin uzun süre saklanması gerekliyse veri arşivlemenin test edilmesi gerekebilir. Arşivleme sırasında belirli verilere ihtiyaç duyulması halinde arşivleme sonrası geri yükleme ve geri alma prosedürlerinin test edilmesi de gerekebilir.

3. Statik Testler – 80 dakika

Anahtar kelimeler

anomali, dinamik test, resmi gözden geçirme, gayri resmi gözden geçirme, teftiş, gözden geçirme, statik analiz, statik test, teknik gözden geçirme, üzerinden geçme

Konu 3 Öğrenme Hedefleri:

3.1 Statik Testin Temelleri

FL-3.1.1 (K1) Farklı statik test teknikleri ile incelenebilecek ürün çeşitlerini ayırt etmek

FL-3.1.2 (K2) Statik testin önemini açıklamak

FL-3.1.3 (K2) Statik ve dinamik testler arasındaki farklılıkları karşılaştırıp kıyaslayabilmek

3.2 Geri Bildirim ve Gözden Geçirme Süreci

FL-3.2.1 (K1) Erken ve sık paydaş geri bildiriminin faydalarını tanımlamak

FL-3.2.2 (K2) Gözden geçirme süreci faaliyetlerini özetlemek

FL-3.2.3 (K1) Gözden geçirme yapılırken ana rollere atanan sorumlulukları hatırlamak

FL-3.2.4 (K2) Farklı gözden geçirme çeşitlerini karşılaştırmak ve ayırt etmek

FL-3.2.5 (K1) Başarılı bir gözden geçirmeye katkıda bulunan faktörleri hatırlamak

3.1. Statik Testin Temelleri

Dinamik testin aksine, statik testte test edilen yazılımın çalıştırılmasına gerek yoktur. Kod, süreç, sistem mimarisi veya diğer çalışma ürünleri manuel incelemeyle (ör. gözden geçirme) veya bir araç yardımıyla (ör. statik analiz) değerlendirilir. Test hedefleri arasında kaliteyi yükseltmek, hataları tespit etmek ve okunabilirlik, bütünlük, doğruluk, test edilebilirlilik ve tutarlılık gibi özellikleri değerlendirmek yer alır. Statik test hem doğrulama hem sağlama için uygulanabilir.

Test uzmanları, iş birimleri ve geliştiriciler örnek eşlemeler, iş birliğine dayalı kullanıcı hikayesi yazma süreci ve iş listesi olgunlaştırma oturumları sırasında birlikte çalışarak kullanıcı hikayelerinin ve ilgili çalışma ürünlerinin tanımlanmış kriterleri [ör. Hazır Tanımı (bkz. bölüm 5.1.3)] karşıladığından emin olurlar. Kullanıcı hikayelerinin tam ve anlaşılır olmasını ve test edilebilir kabul kriterlerini içermesini sağlamak için gözden geçirme teknikleri uygulanabilir. Test uzmanları, doğru soruları sorarak önerilen kullanıcı hikayelerini inceler, sorgular ve iyileştirir.

Statik analiz, dinamik testten önce problemleri belirleyebilir ve test senaryoları gerekmediğinden ve genellikle araçlar (bkz. konu 6) kullanıldığından genel olarak daha az efor gerektirir. Statik analiz genelde CI sürecine dahil edilir (bkz. bölüm 2.1.4). Her ne kadar büyük ölçüde belirli kod hatalarını tespit etmede kullanılsa da statik analiz aynı zamanda sürdürülebilirlik ve güvenliği değerlendirmede de kullanılır. Yazım denetleyici ve okunabilirlik araçları statik analiz araçlarına diğer örneklerdir.

3.1.1. Statik Testlerle İncelenebilen Çalışma Ürünleri

Hemen hemen her çalışma ürünü statik test kullanılarak incelenebilir. Örnekler arasında şunlar yer alır: gereksinimler, kaynak kod, test planları, test senaryoları, ürün iş listesi öğeleri, test başlatma belgeleri, proje dokümantasyonu, sözleşmeler ve modeller.

Okunup anlaşılabilir olan her çalışma ürünü bir gözden geçirmeye tabi olabilir. Bununla birlikte, statik analiz için çalışma ürünlerinin kontrol mekanizması niteliğinde bir yapıya ihtiyacı vardır (ör. modeller, kod veya resmi bir söz dizimine sahip metin).

Statik test için uygun olmayan çalışma ürünleri, insanlar tarafından yorumlanması zor olan ve araçlarla analiz edilemeyecek ürünleri içerir.

3.1.2. Statik Testin Önemi

Statik test YGYD'nün erken aşamalarında hataları tespit ederek erken test prensibini yerine getirebilir (bkz. bölüm 1.3). Ayrıca, dinamik testle tespit edilemeyecek hataları da bulabilir (ör. ulaşılamayan kod, istenildiği gibi uygulanmayan tasarım örüntüleri, çalıştırılamayan kod parçacıklarında yer alan hatalar). Statik testler, çalışma ürünlerinin kalitesini değerlendirmeyi ve çalışma ürünlerine güven duymayı mümkün kılar. Aynı zamanda paydaşlar belgelenmiş gereksinimleri doğrulayarak, bu gereksinimlerin gerçek ihtiyaçlarını tanımladığından emin olabilirler.

Statik testler, YGYD'nün erken aşamalarında yapılabildiği için ilgili paydaşlar arasında ortak bir anlayış oluşturabilir. İlgili paydaşlar arasındaki iletişim de güçlendirilir. Bu nedenle, statik testlere çok çeşitli paydaşların dahil edilmesi önerilir.

Gözden geçirmelerin bir maliyeti olsa da toplam proje maliyeti gözönüne alındığında gözden geçirmelere yapılan yatırımın geri dönüş oranı çok yüksektir. Çünkü projenin ilerleyen aşamalarında hataları düzeltmek için daha fazla zaman ve efor harcanması gerekir.

Kod hataları dinamik test yerine statik analiz kullanılarak daha etkin şekilde tespit edilebilir ve bu da genelde daha az kod hatası ve daha düşük toplam geliştirme eforuyla sonuçlanır.

3.1.3. Statik Test ve Dinamik Test Arasındaki Farklar

Statik test ve dinamik testler birbirlerini tamamlar. Çalışma ürünlerindeki hataların tespitini yardımcı olmak gibi benzer hedefleri olsa da (bkz. bölüm 1.1.1) bazı farklar söz konusudur. Örneğin:

- Statik ve dinamik testlerin (arıza analiziyle) her ikisi de hataların tespit edilmesini sağlayabilir ancak bazı hata çeşitleri ya statik testle ya da dinamik testle bulunabilir.
- Dinamik testler, ilgili hataların daha sonra analiz yoluyla belirleneceği arızalara neden olurken, statik testler hataları doğrudan bulur.
- Statik testler, kod üzerinde nadiren çalıştırılan veya dinamik testlerle zor ulaşılabilen yollar üzerindeki hataları daha kolay tespit edebilir.
- Statik testler çalıştırılamayan alanlara uygulanabilirken, dinamik testler yalnızca çalıştırılabilir alanlara uygulanabilir
- Statik testler kodun çalıştırılmasına bağlı olmayan kalite ölçümünde kullanılabilir (ör. sürdürülebilirlik), dinamik testler ise kodun çalıştırılmasına bağlı olan kalite ölçümünde kullanılabilir (ör. performans)

Statik testler ile bulunması daha kolay ve/veya daha ucuz olan yaygın hatalar şunlardır:

- Gereksinimlerdeki hatalar (ör. tutarsızlıklar, belirsizlikler, çelişkiler, unutulan kısımlar, yanlışlıklar ve tekrarlar)
- Tasarım hataları (ör. verimsiz veritabanı yapıları, zayıf modülerleştirme)
- Kod hatalarının bazı çeşitleri (ör. tanımlanmamış değerlere sahip değişkenler, tanımlanmamış değişkenler, ulaşılamaz veya çoğaltılmış kod, aşırı kod karmaşıklığı)
- Standartlardan sapmalar (ör. kodlama standartlarındaki adlandırma kurallarına bağlı kalmama)
- Hatalı arayüz gereksinimleri (ör. eşleşmeyen parametre sayısı, çeşidi veya sırası)
- Güvenlik açığı çeşitleri (ör. arabellek aşımları)
- Test esası kapsamındaki boşluklar veya yanlışlıklar (ör. bir kabul kriteri için testlerin eksik olması)

3.2. Geri Bildirim ve Gözden Geçirme Süreci

3.2.1. Erken ve Sık Paydaş Geri Bildiriminin Faydaları

Erken ve sık geri bildirimler potansiyel kalite problemlerinin erken iletilmesine olanak sağlar. YGYD boyunca paydaş katılımı çok azsa geliştirilen ürün paydaşın asıl veya mevcut vizyonunu karşılamayabilir. Paydaşın isteklerinin karşılanamaması, masraflı bir yeniden çalışma sürecine, teslim tarihlerinin kaçırılmasına, karşılıklı suçlamalara ve hatta projenin tamamen başarısız olmasına yol açabilir.

YGYD boyunca sıkça verilen paydaş geri bildirimleri gereksinimler hakkındaki yanlış anlaşılmaları önleyebilir ve gereksinimlerdeki değişikliklerin daha erken anlaşılmasını ve uygulanmasını sağlayabilir. Bu da geliştirme ekibinin ne yaptıklarını daha iyi anlamalarına yardımcı olur. Paydaşlara en çok değer katacak ve belirlenen riskler üzerinde en olumlu katkıyı yapacak özelliklere odaklanmalarını sağlar.

3.2.2. Gözden Geçirme Süreci Faaliyetleri

ISO/IEC 20246 standardında, çerçeveleri belli ama duruma göre esnetilebilen yapısal bir gözden geçirme süreci tanımlanmıştır.

Çoğu çalışma ürününün boyutu tek bir gözden geçirmede ele alınamayacak kadar büyüktür. Gözden geçirme süreci, çalışma ürününün tamamının gözden geçirilmesi için birkaç kez tekrarlanabilir.

Gözden geçirme süreci faaliyetleri:

- **Planlama.** Planlama aşamasında amaç, gözden geçirilecek çalışma ürünü, değerlendirilecek kalite karakteristiği, odaklanılacak alanlar, çıkış kriterleri, standartlar, efor gibi destekleyici bilgiler ve gözden geçirmeye ilişkin zaman dilimlerini içeren gözden geçirme kapsamı tanımlanır.
- **Gözden geçirme başlangıcı.** Gözden geçirme başlangıcında amaç, ilgili herkesin ve her şeyin gözden geçirmenin başlatılması için hazır olmasını sağlamaktır. Bu, her katılımcının gözden geçirilen çalışma ürününe erişebilmesini, rollerini ve sorumluluklarını anlamasını ve gözden geçirmenin yapılması için gereken her şeyin karşılanmasını da içerir.
- **Bireysel gözden geçirme.** Her gözden geçirici, gözden geçirilen çalışma ürününün kalitesini değerlendirmek ve bir veya daha fazla gözden geçirme tekniğini uygulayarak (ör. kontrol listesine dayalı gözden geçirme, senaryoya dayalı gözden geçirme) anomalileri, önerileri ve soruları belirlemek için bireysel bir gözden geçirme yapar. ISO/IEC 20246 standardı, farklı gözden geçirme teknikleri hakkında daha fazla bilgi içerir. Gözden geçiriciler belirlenen anomalileri, tavsiyeleri ve soruları kayıt altına alır.
- **İletişim ve analiz.** Gözden geçirme sırasında tespit edilen anomalilerin hata olması gerekmediğinden, tüm bu anomalilerin analiz edilmesi ve tartışılması gerekir. Her anomali için duruma, sahipliğe ve gerekli aksiyonlara göre karar verilmelidir. Bu genellikle, katılımcıların gözden geçirilen çalışma ürününün kalite seviyesine ve hangi takip aksiyonlarının gerekli olduğuna da karar verdikleri bir gözden geçirme toplantısında yapılır. Aksiyonların tamamlanması için bir takip gözden geçirmesi gerekebilir.
- **Düzeltme ve raporlama.** Her hata için bir hata raporu oluşturulmalıdır; böylece düzeltici aksiyonlar takip edilebilir. Çıkış kriterleri sağlandığında çalışma ürünü kabul edilebilir. Gözden geçirme sonuçları raporlanır.

3.2.3. Gözden Geçirmede Roller ve Sorumluluklar

Gözden geçirmeler çeşitli roller üstlenebilecek farklı paydaşlar içerir. Başlıca roller ve sorumluluklar:

- Yönetici, neyin gözden geçirileceğine karar verir ve gözden geçirme için personel ve zaman gibi kaynakları sağlar
- Yazar, gözden geçirilecek çalışma ürününü oluşturur ve düzeltir

- Moderatör (kolaylaştırıcı olarak da bilinir), arabuluculuk, zaman yönetimi ve herkesin özgürce konuşabileceği güvenli bir gözden geçirme ortamı da dahil olmak üzere gözden geçirme toplantılarının etkili bir şekilde yürütülmesini sağlar
- Katip (kaydedici olarak da bilinir), gözden geçiricilerden gelen anomalileri derler ve gözden geçirme toplantısı sırasında alınan kararlar ve bulunan yeni anomaliler gibi gözden geçirme bilgilerini kaydeder
- Gözden geçirici, gözden geçirmeyi gerçekleştirir. Gözden geçirici; projede çalışan herhangi bir kişi olabilir, konu uzmanı veya başka bir paydaş olabilir
- Gözden geçirme lideri, kimin işe dahil olacağına karar vermek ve gözden geçirmenin ne zaman ve nerede yapılacağını organize etmek gibi gözden geçirmenin genel sorumluluğunu üstlenir

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

3.2.4. Gözden Geçirme Çeşitleri

Gayri resmi gözden geçirmelerden resmi gözden geçirmelere kadar birçok gözden geçirme çeşidi bulunur. Gerekli formalite seviyesi, takip edilen YGYD, geliştirme sürecinin olgunluğu, gözden geçirilen çalışma ürününün kritikliği ve karmaşıklığı, yasal veya düzenleyici gereksinimler ve denetim izlemesi ihtiyacı gibi faktörlere dayalıdır. Aynı çalışma ürünü, farklı gözden geçirme çeşitleri kullanılarak gözden geçirilebilir (ör. ilk olarak gayri resmi, ardından resmi gözden geçirme).

Doğru gözden geçirme çeşidini seçmek, gerekli gözden geçirme hedeflerine ulaşmada kilit önem taşır (bkz. bölüm 3.2.5). Seçim sadece hedeflere değil, aynı zamanda proje ihtiyaçları, kullanılabilir kaynaklar, çalışma ürünü çeşidi ve riskleri, iş alanı ve şirket kültürü gibi faktörlere de dayanır.

Yaygın olarak kullanılan gözden geçirme çeşitlerinden bazıları şunlardır:

- **Gayri resmi gözden geçirme.** Gayri resmi gözden geçirmelerde tanımlanmış bir süreç izlenmez ve bunlar resmi olarak belgelenmiş bir çıktı gerektirmez. Ana hedef anomalileri tespit etmektir.
- **Üzerinden geçme.** Yazarın öncülüğündeki bir üzerinden geçme süreci kaliteyi değerlendirmek ve çalışma ürününe yönelik güven oluşturmak, gözden geçiricileri eğitmek, fikir birliğine varmak, yeni fikirler üretmek, yazarları durumları iyileştirmeye ve anomalileri tespit etmeye teşvik etmek ve bunu yapabilmelerini sağlamak gibi birçok amaca hizmet edebilir. Gözden geçiriciler, üzerinden geçme süreci öncesinde bireysel gözden geçirme yapabilir ancak bu gerekli değildir.
- **Teknik Gözden Geçirme.** Teknik gözden geçirmeler teknik açıdan nitelikli gözden geçiriciler tarafından gerçekleştirilir ve bir moderatör tarafından yönetilir. Teknik gözden geçirmenin amaçları arasında teknik bir problemle ilgili fikir birliği sağlamak ve buna ilişkin karar almak, anomalileri tespit etmek, kaliteyi değerlendirmek ve çalışma ürününe yönelik güven oluşturmak, yeni fikirler üretmek, yazarları iyileştirmeler yapmaya teşvik etmek ve buna olanak sağlamak yer alır.
- **Teftiş.** Teftişler en resmi gözden geçirme çeşidi olduğundan tanımlı gözden geçirme süreci eksiksiz olarak izlenir (bkz. bölüm 3.2.2). Ana hedef maksimum sayıda anomali bulmaktır. Diğer hedefler arasında kaliteyi değerlendirmek, çalışma ürününe yönelik güven oluşturmak, yazarları iyileştirmeler yapmaya teşvik etmek ve bunları yapabilmelerini sağlamak bulunur. Metrikler toplanır ve teftiş süreci de dahil olmak üzere YGYD'nü iyileştirmek için kullanılır. Teftişte yazar; gözden geçirme lideri veya katip olarak görev yapamaz.

3.2.5. Gözden Geçirmelerin Başarı Faktörleri

Gözden geçirmelerin başarısını belirleyen çeşitli faktörler vardır ve bunların arasında şunlar yer alır:

- Net hedeflerin ve ölçülebilir çıkış kriterlerinin tanımlanması.
- Verilen hedeflere uygun olan ve çalışma ürününün çeşidine, gözden geçirme katılımcılarına, proje ihtiyaçlarına ve bağlamına uyacak şekilde gözden geçirme türünün seçilmesi
- Gözden geçiriciler bireysel gözden geçirme ve/veya (yapılıyorsa) gözden geçirme toplantısı sırasında konsantrasyonlarını kaybetmesinler diye gözden geçirmelerin küçük parçalar üzerinde gerçekleştirilmesi
- Ürünü ve aktivitelerini iyileştirebilmeleri için paydaşlara ve yazarlara gözden geçirmelerden geri bildirim sağlanması (bkz. bölüm 3.2.1)
- Gözden geçirmeye hazırlanmaları için katılımcılara yeterli sürenin verilmesi
- Gözden geçirme süreci için yönetimden destek sağlanması
- Öğrenme ve süreç iyileştirmesini desteklemek için gözden geçirmelerin şirket kültürünün bir parçası haline getirilmesi
- Rollerini yerine getirmelerini sağlamak için tüm katılımcılara yeterli eğitimin verilmesi
- Kolaylaştırıcı toplantılar

4. Test Analizi ve Tasarımı - 390 dakika

Anahtar kelimeler

kabul kriterleri, kabul testi güdümlü yazılım geliştirme, kara kutu test tekniği, sınır değer analizi, dal kapsamı, kontrol listesine dayalı test etme, iş birliğine dayalı test yaklaşımı, kapsam, kapsam öğesi, karar tablosu testi, denklik paylarına ayırma, hata tahminleme, tecrübeye dayalı test tekniği, keşif testi, durum geçişi testi, komut kapsama yüzdesi, test tekniği, beyaz kutu test tekniği

Konu 4 Öğrenme Hedefleri:

4.1 Test Tekniklerine Genel Bakış

FL-4.1.1 (K2) Kara kutu, beyaz kutu ve tecrübeye dayalı test teknikleri arasındaki farkı ayırt etmek

4.2 Kara Kutu Test Teknikleri

FL-4.2.1 (K3) Test senaryoları elde etmek için denklik paylarına ayırma test tekniğini kullanmak

FL-4.2.2 (K3) Test senaryoları elde etmek için sınır değer analizi test tekniğini kullanmak

FL-4.2.3 (K3) Test senaryoları elde etmek için karar tablosu test tekniğini kullanmak

FL-4.2.4 (K3) Test senaryoları elde etmek için durum geçişi test tekniğini kullanmak

4.3 Beyaz Kutu Test Teknikleri

FL-4.3.1 (K2) Komut test tekniğini açıklamak

FL-4.3.2 (K2) Dal test tekniğini açıklamak

FL-4.3.3 (K2) Beyaz kutu testinin önemini açıklamak

4.4 Tecrübeye Dayalı Test Teknikleri

FL-4.4.1 (K2) Hata tahminlemeyi açıklamak

FL-4.4.2 (K2) Keşif testini açıklamak

FL-4.4.3 (K2) Kontrol listesine dayalı test tekniğini açıklamak

4.5. İş Birliğine Dayalı Test Yaklaşımları

FL-4.5.1 (K2) Geliştiriciler ve iş birimleri ile iş birliği içinde nasıl kullanıcı hikayeleri yazılacağını açıklamak

FL-4.5.2 (K2) Kabul kriteri yazımı için farklı seçenekleri sınıflandırmak

FL-4.5.3 (K3) Test senaryoları elde etmek için kabul testi güdümlü geliştirmeyi (ATDD) kullanmak

4.1. Test Tekniklerine Genel Bakış

Test teknikleri, test analizinde (ne test edilecek) ve test tasarımında (nasıl test edilecek) test uzmanına destek sağlar. Test teknikleri, nispeten az ancak yeterli sayıda test senaryosunun sistematik bir şekilde geliştirilmesine yardımcı olur. Test teknikleri ayrıca test uzmanının test analizi ve tasarımı sırasında test koşullarını tanımlamasına, kapsam öğelerini belirlemesine ve test verilerini tanımlamasına yardımcı olur. Test teknikleri ve ilgili ölçülere ilişkin ayrıntılı bilgiye ISO/IEC/IEEE 29119-4 standardında ve (Beizer 1990, Craig 2002, Copeland 2004, Koomen 2006, Jorgensen 2014, Ammann 2016, Forgács 2019) kaynaklarında da yer verilmiştir.

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

Kara kutu test teknikleri (spesifikasyon bazlı teknikler olarak da bilinir) test nesnesinin iç yapısına atıfta bulunulmadan test nesnesinin davranışının analiz edilmesine dayanır. Bu nedenle test senaryoları yazılımın nasıl yazıldığından bağımsızdır. Sonuç olarak, kod değişir ancak test nesnesinin davranışı aynı kalırsa test senaryoları da hâlâ faydalı demektir.

Beyaz kutu test teknikleri (yapı bazlı teknik olarak da bilinir) test nesnesinin iç yapısının ve işlemlerinin analizine dayanır. Test senaryoları yazılımın nasıl tasarlandığına bağlı olduğundan sadece test nesnesinin tasarımı veya kodlanmasından sonra oluşturulabilir.

Tecrübeye dayalı test teknikleri test senaryolarının tasarımı ve uyarlanması süreci için test uzmanlarının bilgi ve deneyimlerini etkin bir şekilde kullanır. Bu tekniklerin etkinliği ağırlıklı olarak test uzmanının becerilerine dayalıdır. Tecrübeye dayalı test teknikleri kara kutu ve beyaz kutu test tekniklerinde gözden kaçırılabilecek hataları tespit edebilir. Dolayısıyla tecrübeye dayalı test teknikleri kara kutu ve beyaz kutu test tekniklerini tamamlayıcı niteliktedir.

4.2. Kara Kutu Test Teknikleri

Yaygın olarak kullanılan kara kutu test teknikleri aşağıdaki bölümlerde ele alınmaktadır:

- Denklik Paylarına Ayırma
- Sınır Değer Analizi
- Karar Tablosu Testleri
- Durum Geçişi Testleri

4.2.1. Denklik Paylarına Ayırma

Denklik Paylarına Ayırma (DPA), belirli bir payın tüm unsurlarının test nesnesi tarafından aynı şekilde işleneceği beklentisine dayanarak verileri paylara (denklik payları olarak bilinir) ayırır. Bu tekniğin ardındaki teori, denklik payından bir değeri test eden bir test senaryosunun bir hata tespit etmesi halinde bu hatanın aynı paydan başka herhangi bir değeri test eden test senaryoları tarafından da tespit edilmesi gerektiği yaklaşımına dayanır. Dolayısıyla her pay için bir test yapılması yeterlidir.

Denklik payları; girdiler, çıktılar, yapılandırma öğeleri, dahili değerler, zamana bağlı değerler ve arayüz parametreleri dahil olmak üzere test nesnesiyle ilgili herhangi bir veri öğesi için tanımlanabilir. Paylar sürekli veya ayrık, sıralı veya sırasız, sonlu veya sonsuz olabilir. Paylar birbiriyle çakışmamalı ve boş küme halinde olmamalıdır.

Basit test nesneleri için DPA tekniğini kullanmak kolay olabilir ancak pratikte test nesnesinin farklı

değerlerde nasıl davranacağını anlamak genelde karmaşıktır. Dolayısıyla paylara ayırma titizlikle yapılmalıdır.

Geçerli değerler içeren paylara geçerli pay denir. Geçersiz değerler içeren paylara geçersiz pay denir. Geçerli ve geçersiz değerlerin tanımları ekipler ve kuruluşlar arasında değişiklik gösterebilir. Örneğin; geçerli değerler, test nesnesi tarafından işlenmesi gereken veya gereksinimlerin bunların işlenmesini tanımladığı değerler olarak yorumlanabilir. Geçersiz değerler, test nesnesi tarafından göz ardı edilmesi veya reddedilmesi gereken değerler veya test nesnesi gereksiniminde hiçbir işlem tanımlamasına tabi tutulmayan değerler olarak yorumlanabilir.

DPA'da kapsam öğeleri denklik paylarıdır. Bu teknikle kapsamın %100'üne ulaşmak için test senaryoları, her payı en az bir kez kapsama alarak, belirlenmiş tüm payları (geçersiz paylar dahil) denemesi gerekmektedir. Kapsam, denenen payların sayısının, tanımlanan payların toplam sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

Çoğu test nesnesi birden fazla pay grubu içerir (ör. birden fazla girdi parametresine sahip test nesneleri); bu da bir test senaryosunun farklı pay gruplarından paylar kapsayacağı anlamına gelir. Birden fazla pay grubu olması durumunda en basit kapsam kriteri Each Choice kapsamı (Her Seçeneği Kapsama) olarak adlandırılır (Ammann 2016). Each Choice kapsamı, test senaryolarının her pay grubundan her bir payı en az bir kez denemesini gerektirir. Each Choice kapsamı pay kombinasyonlarını dikkate almaz.

4.2.2. Sınır Değer Analizi

Sınır Değer Analizi (SDA) denklik paylarının sınırlarının denenmesine dayalı bir tekniktir. Dolayısıyla SDA sadece sıralı verilerden oluşan paylarda kullanılabilir. Bir payın minimum ve maksimum değerleri o payın sınır değerleridir.

SDA payların sınır değerlerine odaklanır çünkü geliştiricilerin bu sınır değerlerde hata yapması daha olasıdır. SDA tarafından bulunan tipik hatalar, sınırların istenen değerlerin üzerinde veya altında değerler olarak yanlış uygulandığı veya tamamen ihmal edildiği durumlarda görülmektedir.

Bu ders programı SDA'nın iki versiyonunu içerir: 2 değerli ve 3 değerli SDA. Bunlar %100 kapsama ulaşmak için denenmesi gereken sınır başına kapsam öğeleri açısından farklılık gösterir.

2 değerli SDA'da (Craig 2002, Myers 2011) her sınır değer için iki kapsam öğesi vardır. Bunlar: sınır değer ve bitişik paya ait en yakın komşusu. 2 değerli SDA ile %100 kapsama ulaşmak için test senaryolarının tüm kapsam öğelerini, yani tanımlanan tüm sınır değerlerini denemesi gerekir. Kapsam, test senaryolarında denenen sınır değerlerin sayısının tanımlanan tüm sınır değerlerin sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

3 değerli SDA'da (Koomen 2006, O'Regan 2019) her sınır değer için üç kapsam öğesi vardır. Bunlar: sınır değer ve bu sınır değerin her iki komşu sınır değeri. Dolayısıyla 3 değerli SDA'da bazı kapsam öğeleri sınır değerler olmayabilir. 3 değerli SDA ile %100 kapsama ulaşmak için test senaryolarının tüm kapsam öğelerini, yani tanımlanan sınır değerleri ve komşu değerlerini denemesi gerekir. Kapsam, denenen sınır değerlerin ve komşularının sayısının tanımlanan tüm sınır değerlerin ve komşularının sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

3 değerli SDA, 2 değerli SDA'dan daha titiz bir analizdir çünkü 3 değerli SDA 2 değerli SDA'nın gözden kaçırdığı hataları tespit edebilir. Örneğin "(x ≤ 10)" olması gerekirken "(x = 10) " olarak hatalı şekilde kodlandıysa 2 değerli SDA'dan (x = 10, x = 11) elde edilen hiçbir test verisi hatayı bulamaz. Ancak 3 değerli SDA'dan elde edilen x = 9 hatayı bulabilir.

4.2.3. Karar Tablosu Testleri

Karar tabloları, farklı test koşul kombinasyonlarının nasıl farklı çıktılar ürettiğini göstermek için kullanılır. Karar tabloları, ayrıca iş kuralları gibi karmaşık mantığa sahip bileşenleri göstermenin etkin bir yoludur.

Karar tabloları oluşturulurken sistemin koşulları ve bunların sonucunda ortaya çıkan aksiyonlar tanımlanır. Bunlar tablonun satırlarını oluşturur. Her sütun, koşulların benzersiz bir kombinasyonunu ve ilişkili aksiyonları tanımlayan bir karar kuralına karşılık gelir.

Sınırlı giriş karar tablolarında koşul ve aksiyonların tüm değerleri (ilgisiz veya uygulanabilir olmayan hariç; aşağıya bakın) Boolean (doğru veya yanlış) olarak gösterilir. Genişletilmiş giriş karar tablolarında ise koşul ve aksiyonların bazıları veya tamamı birden fazla değer alabilir (ör. sayı aralıkları, denklik payları, ayrık değerler).

Koşul gösterimi şu şekildedir: "T" (doğru) koşulun yerine getirildiği anlamına gelir. "F" (yanlış) koşulun yerine getirilmediği anlamına gelir. "–", koşulun değerinin aksiyon çıktısı açısından ilgisiz olduğu anlamına gelir. "N/A – Not Applicable" koşulun verilen kural için uygulanabilir olmadığı anlamına gelir. Aksiyonlar için: "X", aksiyonun gerçekleşmesi gerektiği anlamına gelir. Boşluk, aksiyonun gerçekleşmemesi gerektiği anlamına gelir. Başka gösterim şekilleri de kullanılabilir.

Tam bir karar tablosunda her koşul kombinasyonunu kapsayacak kadar sütun vardır. Tablo, elverişsiz koşul kombinasyonları içeren sütunların silinmesiyle sadeleştirilebilir. Tablo, bazı koşulların çıktıyı etkilemeyeceği sütunların tek bir sütunda birleştirilmesiyle de küçültülebilir. Karar tablosu küçültme algoritmaları (indirgenmiş karar tabloları) bu ders programı kapsamında değildir.

Karar tablosu testinde, kapsam öğeleri, koşulların uygulanabilir kombinasyonlarını içeren sütunlardır.

Bu teknikle %100 kapsama ulaşmak için test senaryoları tüm bu sütunları denemelidir. Kapsam, denenen sütunların sayısının tüm uygulanabilir sütunların sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

Karar tablosu testlerinin güçlü yanı, hiçbir koşulu göz ardı etmeden tüm koşul kombinasyonlarını ele almaya yönelik sistematik bir yaklaşım sağlamasıdır. Ayrıca gereksinimlerdeki boşlukları veya çelişkileri bulmaya da yardımcı olur.

Koşulların sayısı fazla olduğunda, karar kurallarının tümünü uygulamak zaman alabilir, çünkü kuralların sayısı koşulların sayısıyla üssel olarak artar. Bu durumda, uygulanması gereken kural sayısını azaltmak için indirgenmiş bir karar tablosu veya risk temelli bir yaklaşım kullanılabilir.

4.2.4. Durum Geçişi Testleri

Durum geçişi diyagramları, bir sistemin davranışını sistemin olası durumlarını ve geçerli durum geçişlerini göstererek modeller. Bir geçiş, bir koruma koşulu ile ilişkilendirilebilecek bir olayla başlatılır. Geçişlerin anlık olduğu varsayılır ve bazen bu geçişler yazılımla ilgili bir aksiyonla sonuçlanır. Yaygın kullanım şu şekildedir: "olay [koruma koşulu] / aksiyon". Koruma koşulları ve aksiyonlar mevcut değilse veya test uzmanı açısından alakasız ise göz ardı edilebilir.

Durum tablosu, durum geçişi diyagramı ile eş değer bir modeldir. Tablonun satırları durumları, sütunları ise olayları temsil eder (varsa koruma koşullarıyla birlikte). Tablo girişleri (hücreler) geçişleri temsil eder ve hedef durum ve tanımlanmışsa sonuç olarak ortaya çıkan aksiyonları içerir.

Durum geçişi diyagramının aksine, durum tablosu geçersiz geçişleri açıkça gösterir, bunlar boş hücrelerle temsil edilir.

Durum geçişi diyagramına veya durum tablosuna dayalı bir test senaryosu genelde, bir dizi durum değişikliğiyle (ve gerekirse aksiyonla) sonuçlanan bir dizi olayla temsil edilir. Bir test senaryosu, durumlar arası çeşitli geçişleri içerebilir.

Durum geçişi testi için birçok kapsama kriteri vardır. Bu ders programında bunlardan üçü ele alınmıştır.

Tüm **durumlar kapsamında** kapsam öğesi durumlardır. Tüm durumlar kapsamının %100'üne ulaşmak için test senaryoları tüm durumların incelenmesini sağlamalıdır. Kapsam, incelenen durum sayısının toplam durum sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

Geçerli geçişler kapsamında (0-anahtar kapsamı da denir) kapsam öğeleri tekil geçerli geçişlerdir. Geçerli geçişler kapsamının %100'üne ulaşmak için test senaryoları tüm geçerli geçişleri denemelidir. Kapsam, denenen geçerli geçişlerin sayısının toplam geçerli geçiş sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

Tüm geçişler kapsamında kapsam öğeleri durum tablosunda gösterilen tüm geçişlerdir. Tüm geçişler kapsamının %100'üne ulaşmak için test senaryoları tüm geçerli geçişleri denemeli ve geçersiz geçişleri denemeye çalışmalıdır. Tek bir test senaryosunda yalnızca bir geçersiz geçişin test edilmesi, kusur maskelenmesini, yani bir hatanın diğerinin tespitini engellediği durumları önlemeye yardımcı olur. Kapsam, yürütülen test senaryoları tarafından denenen veya kapsanmaya çalışılan geçerli ve geçersiz geçiş sayısının toplam geçerli ve geçersiz geçiş sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

Tüm durumlar kapsamı, geçerli geçişler kapsamından daha zayıftır çünkü genelde bu kapsama tüm geçişler denenmeden ulaşılabilir. Geçerli geçişler kapsamı en yaygın kullanılan kapsama kriteridir. Geçerli geçişler kapsamının tamamına ulaşmak, tüm durumlar kapsamının tamamına ulaşılmasını garanti eder. Tüm geçişler kapsamına ulaşmak, hem tüm durumlar kapsamının hem de geçerli geçişler kapsamının tamamına ulaşılmasını garanti eder; aynı zamanda kritik ve emniyet hassasiyetli yazılımlar için minimum gereksinim niteliğinde olmalıdır.

4.3. Beyaz Kutu Test Teknikleri

Hem yaygın olarak kullanılmaları hem de basit olmaları nedeniyle bu bölüm koda bağlı iki adet beyaz kutu test tekniğine odaklanır:

- Komut testi
- Dal testi

Daha eksiksiz bir kod kapsamı elde etmek için bazı emniyet hassasiyetli, kritik veya çok fazla entegrasyonun olduğu ortamlarda kullanılan daha kesin teknikler de vardır. Daha yüksek test seviyelerinde kullanılan (ör. API testi) veya koda bağlı olmayan kapsamın kullanıldığı (ör. nöral ağ testindeki nöron kapsamı) beyaz kutu test teknikleri de bulunur. Bu tekniklere bu ders programında yer verilmemiştir.

4.3.1. Komut Testleri ve Komut Kapsama Yüzdesi

Komut testinde kapsam öğeleri çalıştırılabilir komutlardır. Amaç, kabul edilebilir bir kapsama seviyesi elde edilene kadar kod içindeki komutları deneyecek test senaryoları tasarlamaktır. Kapsam, test senaryoları tarafından denenen komut sayısının kod içindeki toplam çalıştırılabilir komut sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

%100 komut kapsama yüzdesine ulaşıldığında kod içindeki tüm çalıştırılabilir komutların en az bir kez denenmesi sağlanır. Bu özellikle, hata içeren her bir komutun yürütüleceği anlamına gelir ve bu durum, hatanın varlığını gösteren bir arıza durumuna neden olabilir. Ancak, bir komutun denenmesi demek, bu komutla ilgili tüm hataların tespit edileceği anlamına gelmez. Örneğin, veriye bağımlı hatalar tespit

edilemeyebilir (ör. yalnızca payda sıfır olarak ayarlandığında başarısız olan bir sıfıra bölme işlemi). Bunun yanı sıra, %100 komut kapsama yüzdesi tüm karar mantığının test edilmesini sağlamaz çünkü kod içindeki tüm dallar (bkz. konu 4.3.2) denenmeyebilir.

4.3.2. Dal Testi ve Dal Kapsamı

Dal, test nesnesinde komutların çalıştırıldığı olası sıraları gösteren kontrol akış grafiğindeki iki düğüm arasında gerçekleşen bir kontrol transferidir. Her bir kontrol transferi koşulsuz (ör. doğrudan açık kod) veya koşullu (ör. karar çıktısı) olabilir.

Dal testlerinde kapsam öğeleri dallardır ve burada amaç, kabul edilebilir bir kapsama seviyesi elde edilene kadar kod içindeki dalları deneyecek test senaryoları tasarlamaktır. Kapsam, test senaryoları tarafından denenen dal sayısının toplam dal sayısına bölünmesiyle ölçülür ve yüzde olarak ifade edilir.

%100 dal kapsamına ulaşıldığında, kod içindeki koşulsuz ve koşullu tüm dallar test senaryoları tarafından denenir. Koşullu dallar tipik olarak bir "if...then" kararının true veya false çıktısına, bir switch/case komutunun çıktısına veya bir döngüden çıkma veya döngüye devam etme kararına karşılık gelir. Ancak, bir dalın bir test senaryosuyla denenmesi, o daldaki tüm hataların tespit edileceği anlamına gelmez. Örneğin, bir kodda spesifik bir yolun yürütülmesini gerektiren hataları tespit edemeyebilir.

Dal kapsamı, komut kapsama yüzdesini içerir. Bu, dal kapsamının %100'üne ulaşan her test senaryosu grubunun aynı zamanda %100'lük bir komut kapsama yüzdesine ulaşacağı anlamına gelir (ancak bunun tersi söz konusu değildir).

4.3.3. Beyaz Kutu Testinin Önemi

Tüm beyaz kutu test tekniklerinin başlıca kuvvetli yanı, test sırasında kodun tamamının dikkate alınmasıdır; bu da analiz belirsiz, eski veya eksik olsa bile hata tespitini kolaylaştırır. Beyaz kutu testinin sadece kodu baz alarak testleri koşturmasının ortaya çıkardığı zafiyet ise kodlanmayan gereksinimler varsa bunları yakalayamayacak olmasıdır. (Watson 1996).

Beyaz kutu test teknikleri statik testte kullanılabilir (ör. kodun deneme koşumları sırasında). Kullanılabileceği alanlar, henüz koşum için hazır olmayan kod, (Hetzel 1988) kaba kod ve bir kontrol akış grafiği ile modellenebilen yazılımlar vb.

Sadece kara kutu testi gerçekleştirmek, gerçek kod kapsamının ölçülmesini sağlamaz. Beyaz kutu kapsam ölçüleri objektif bir kapsam ölçümü sağlar ve bu kapsamı büyütmek için üretilecek ek testlere olanak sağlayan gerekli bilgileri sunar ve dolayısıyla koda yönelik güveni artırır.

4.4. Tecrübeye Dayalı Test Teknikleri

Yaygın olarak kullanılan tecrübeye dayalı test teknikleri aşağıdaki bölümlerde ele alınmaktadır:

- Hata tahminleme
- Keşif testi
- Kontrol listesine dayalı test

4.4.1. Hata Tahminleme

Hata tahminleme, test uzmanının bilgilerine dayalı olarak insan hatalarının, yazılım hatalarının ve arızaların ortaya çıkmasını sağlamak için kullanılan bir tekniktir; bu bilgiler aşağıdaki gibidir:

- Uygulamanın geçmişte nasıl çalıştığı

- Geliştiricilerin yapma eğiliminde olduğu kusurlar ve bu kusurların sonucu olarak meydana gelen hata çeşitleri
- Diğer benzer uygulamalarda oluşan arıza çeşitleri

Genel olarak insan hataları, yazılım hataları ve arızalar şunlarla ilgili olabilir: girdi (ör. doğru girdinin kabul edilmemesi, yanlış veya eksik parametreler), çıktı (ör. yanlış format, yanlış sonuç), mantık (ör. eksik senaryolar, yanlış operatör), hesaplama (ör. yanlış işlenen, yanlış hesaplama), arayüz (ör. parametre uyuşmazlığı, uyumsuz türler) veya veriler (ör. yanlış öndeğer atama, yanlış tür).

Kusur ortaya çıkarmaya yönelik saldırılar hata tahminlemenin uygulanmasına yönelik metodik bir yaklaşımdır. Bu teknikte test uzmanının olası insan hatalarının, yazılım hatalarının ve arızaların bir listesini oluşturması veya elde etmesi ve insan hatalarıyla ilişkili hataları tanımlayacak, hataları ortaya çıkaracak veya arızalara neden olacak testler tasarlaması gerekir. Bu listeler tecrübeye, hata ve arıza verilerine dayanarak veya yazılımların neden başarısız olduğu hakkındaki genel bilgilerden yola çıkarak oluşturulabilir.

Hata tahminleme ve kusur ortaya çıkarmaya yönelik saldırılar hakkında daha fazla bilgi için bkz. (Whittaker 2002, Whittaker 2003, Andrews 2006).

4.4.2. Keşif Testi

Keşif testinde, test uzmanı test nesnesi hakkında bilgi edinirken testler eş zamanlı olarak tasarlanır, koşulur ve değerlendirilir. Test, test nesnesi hakkında daha fazla bilgi sahibi olmak, odak testlerle test nesnesini daha derinlemesine keşfetmek ve test edilmemiş alanlara yönelik testler oluşturmak için kullanılır.

Keşif testi sürecini daha sistematik yapmak için oturum bazlı testlerden faydalanılır. Oturum bazlı yaklaşımda, keşif testi belirli bir zaman dilimi içinde yapılır. Test uzmanı, test sürecine rehberlik etmesi için test hedeflerini içeren bir test başlatma belgesi kullanır. Test oturumunun ardından genellikle test uzmanı ile test oturumunun sonuçlarıyla ilgilenen paydaşların görüşmesini içeren bir bilgilendirme toplantısı yapılır. Bu yaklaşımda test hedefleri üst seviye test koşulları olarak görülebilir. Kapsam öğeleri test oturumunda tanımlanır ve denenir. 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 işe yarar. Keşif testleri, diğer daha resmi test tekniklerini tamamlamak için de kullanılır. Test uzmanı deneyimli, alan bilgisine sahip ve analitik beceriler, merak ve yaratıcılık gibi üst seviye temel becerilere sahipse keşif testleri daha etkili olacaktır (bkz. bölüm 1.5.1).

Keşif testleri diğer test tekniklerinin kullanımını içerebilir (ör. denklik paylarına ayırma). Keşif testleri hakkında daha fazla bilgi için bkz. (Kaner 1999, Whittaker 2009, Hendrickson 2013).

4.4.3. Kontrol Listesine Dayalı Testler

Kontrol listesine dayalı testlerde, test uzmanları bir kontrol listesinde bulunan test koşullarını kapsayacak şekilde testler tasarlar, uygular ve koşar. Kontrol listeleri tecrübeye, kullanıcı için neyin önemli olduğu veya yazılımın niçin ve nasıl başarısız olabileceği bilgisine dayanarak oluşturulabilir. Kontrol listeleri otomatik olarak kontrol edilebilecek öğeleri, daha çok giriş/çıkış kriteri olmaya uygun olan öğeleri veya çok genel olan öğeleri içermez (Brykczynski 1999).

Kontrol listesi öğeleri genellikle soru şeklindedir. Her öğe ayrı ayrı ve doğrudan kontrol edilebilir olmalıdır. Bu öğeler gereksinimlere, arayüz özelliklerine, kalite karakteristiğine veya diğer test koşulu biçimlerine ilişkin olabilir. Fonksiyonel ve fonksiyonel olmayan testler de dahil olmak üzere farklı test çeşitlerini desteklemek için kontrol listeleri oluşturulabilir [ör. kullanılabilirlik testi için 10 kriter (Nielsen 1994)].

Bazı kontrol listeleri zaman içinde kademeli olarak daha az etkin hale gelebilir çünkü geliştiriciler aynı hataları yapmaktan kaçınmayı öğrenecektir. Yeni bulunan yüksek önem derecesine sahip hataların da yansıtılması için yeni kriterlerin eklenmesi gerekebilir. Dolayısıyla kontrol listeleri hata analizine dayalı şekilde düzenli olarak güncellenmelidir. Bununla birlikte, kontrol listesinin çok uzun olmamasına dikkat edilmelidir (Gawande 2009).

Ayrıntılı test senaryolarının olmadığı durumlarda, kontrol listesine dayalı testler, test süreci için yol gösterici olabilir. Bu kontrol listeleri 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.

4.5. İş Birliğine Dayalı Test Yaklaşımları

Yukarıdaki tekniklerden her biri (bkz. bölüm 4.2, 4.3, 4.4) hata tespitiyle ilgili özel bir hedefe hizmet eder. İş birliğine dayalı yaklaşımlar ise hataların iş birliği ve iletişim yoluyla önlenmesine odaklanır.

4.5.1. İş Birliğine Dayalı Kullanıcı Hikayesi Yazımı

Kullanıcı hikayeleri, bir sistem ya da yazılımın kullanıcısı veya alıcısı için değerli olacak bir özelliği temsil eder. Kullanıcı hikayelerinin üç kritik unsuru bulunur (Jeffries 2000); bunlara birlikte "3 C'ler" denir:

- Card (Kart): Bir kullanıcı hikayesini açıklayan ortam (ör. dizin kartı, elektronik panodaki bir giriş)
- Conversation (Konuşmalar): Yazılımın nasıl kullanılacağını açıklar (belgeyle veya sözlü olabilir)
- Confirmation (Onay): Kabul kriterleri (bkz. bölüm 4.5.2)

Bir kullanıcı hikayesi için en yaygın format "[Rol] olarak [gerçekleştirilecek hedef] istiyorum çünkü [rol için ortaya çıkan iş değeri]." şeklindedir ve bu formatı kabul kriterleri izler.

İş birlikçi kullanıcı hikayesi yazımı sırasında beyin fırtınası ve zihin haritası gibi teknikler kullanılabilir. İş birliği, ekibin analiz, yazılım geliştirme ve test etme gibi üç perspektifi dikkate alarak, neyin teslim edilmesi gerektiğine yönelik ortak bir vizyon edinmesini sağlar.

İyi kullanıcı hikayeleri: Bağımsız, Müzakere edilebilir, Değerli, Tahmin edilebilir, Küçük ve Test edilebilir (INVEST) olmalıdır. Paydaş bir kullanıcı hikayesinin nasıl test edileceğini bilmiyorsa bu durum kullanıcı hikayesinin yeterince net olmadığını veya kendisi için değerli bir şeyi yansıtmadığını veya paydaşın sadece test konusunda yardıma ihtiyacı olduğunu işaret eder (Wake 2003).

4.5.2. Kabul Kriterleri

Bir kullanıcı hikayesi için kabul kriterleri, bu kullanıcı hikayesinin, paydaşlar tarafından kabul edilmesi için karşılaması gereken koşulları içerir. Bu bakış açısından bakıldığında kabul kriterleri, test uzmanları tarafından denenmesi gereken test koşulları olarak görülebilir. Kabul kriterleri genelde bir konuşma sonunda ortaya çıkar (bkz. bölüm 4.5.1).

Kabul kriterleri şu amaçlarla kullanılır:

- Kullanıcı hikayesinin kapsamını tanımlamak
- Paydaşlar arasında fikir birliğini sağlamak
- Hem pozitif hem negatif senaryoları açıklamak
- Kullanıcı hikayesi kabul testi için esas teşkil etmek (bkz. bölüm 4.5.3)

- Doğru planlama ve tahminleme yapılmasını sağlamak

Bir kullanıcı hikayesi için kabul kriterleri yazmanın birçok yolu vardır. En yaygın iki format:

- Senaryo odaklı (ör. BDD'de kullanılan Given/When/Then formatı, bkz. bölüm 2.1.3)
- Kural odaklı (ör. madde işaretli kontrol listesi veya girdi-çıktı eşlemesinin tablo haline getirilmiş şekli)

Çoğu kabul kriteri bu iki formattan birinde dokümante edilebilir. Ancak, kabul kriterleri iyi tanımlanmış ve açık olduğu sürece ekip başka bir özel format da kullanabilir.

4.5.3. Kabul Testi Güdümlü Yazılım Geliştirme (ATDD)

ATDD bir önce-test-et yaklaşımıdır (bkz. bölüm 2.1.3). Kullanıcı hikayesini uygulamadan önce test senaryoları oluşturulur. Test senaryoları farklı perspektiflere sahip ekip üyeleri tarafından oluşturulur (ör. müşteriler, geliştiriciler ve test uzmanları) (Adzic 2009). Test senaryoları manuel veya otomatik olarak yürütülebilir.

İlk adım, kullanıcı hikayesinin ve (henüz tanımlanmamışsa) kabul kriterlerinin ekip üyeleri tarafından analiz edildiği, görüşüldüğü ve yazıldığı bir analiz çalıştayıdır. Kullanıcı hikayesindeki tüm eksiklikler, belirsizlikler veya hatalar bu süreçte düzeltilir. Bir sonraki adım, test senaryoları oluşturmaktır. Bu, ekip tarafından bir bütün olarak veya test uzmanı tarafından bireysel olarak yapılabilir. Test senaryoları kabul kriterlerine bağlıdır ve yazılımın nasıl çalıştığına ilişkin örnekler olarak görülebilir. Bu, ekibin kullanıcı hikayesini doğru hayata geçirmesine yardımcı olur.

Örnekler ve testler aynı olduğundan, bu terimler sıkça birbirinin yerine kullanılır. Test tasarımı sırasında bölüm 4.2, 4.3 ve 4.4'te açıklanan test teknikleri uygulanabilir.

Tipik olarak, ilk test senaryoları, istisnalar veya hata koşulları olmaksızın doğru davranışı onaylayan ve her şey beklendiği gibi gittiğinde yürütülen aktivite dizisini oluşturan pozitif testlerdir. Pozitif test senaryolarından sonra ekip negatif testler yapmalıdır. Son olarak ekip fonksiyonel olmayan gereksinimleri de ele almalıdır (ör. performans, kullanılabilirlik). Test senaryoları paydaşlar için anlaşılır olacak şekilde ifade edilmelidir. Genelde test senaryoları gerekli önkoşulları (varsa), girdileri ve artkoşulları içeren doğal bir dilde cümleler içerir.

Test senaryoları kullanıcı hikayesinin tüm karakteristiklerini ele almalı ve hikayenin ötesine geçmemelidir. Ancak kabul kriterleri, kullanıcı hikayesinde belirtilen sorunlardan bazılarını detaylandırabilir. Ayrıca, iki test senaryosu kullanıcı hikayesinin aynı karakteristiklerini belirtmemelidir.

Test senaryoları bir test otomasyon çerçevesiyle desteklenen bir formatta olduğunda, geliştiriciler bir kullanıcı hikayesinde tanımlanan özelliği uygularken destekleyici kodu yazarak test senaryolarını otomatik hale getirebilir. Böylece kabul testleri, koşturulabilir gereksinimler haline gelir.

5. Test Aktivitelerini Yönetme – 335 dakika

Anahtar kelimeler

hata yönetimi, hata raporu, giriş kriterleri, çıkış kriterleri, ürün riski, proje riski, risk, risk analizi, risk değerlendirmesi, risk kontrolü, risk belirleme, risk seviyesi, risk yönetimi, risk azaltma, risk gözetimi, risk bazlı test, test yaklaşımı, test tamamlama raporu, test kontrol, test gözetimi, test planı, test planlama, test ilerleme raporu, test piramidi, test çeyrekleri

Konu 5 Öğrenme Hedefleri:

5.1 Test Planlama

FL-5.1.1 (K2) Bir test planının amacına ve içeriğine ilişkin örnekler vermek

FL-5.1.2 (K1) Bir test uzmanının döngü ve sürüm planlamasına nasıl değer kattığını anlamak

FL-5.1.3 (K2) Giriş kriterleri ve çıkış kriterlerini karşılaştırıp kıyaslamak

FL-5.1.4 (K3) Gerekli test eforunu hesaplamak için tahminleme teknikleri kullanmak

FL-5.1.5 (K3) Test senaryosularını önceliklendirmek

FL-5.1.6 (K1) Test piramidinin konseptlerini hatırlamak

FL-5.1.7 (K2) Test çeyreklerini ve bunların test seviyeleri ve test çeşitleri ile olan ilişkilerini özetlemek

5.2 Risk Yönetimi

FL-5.2.1 (K1) Risk olasılığı ve risk etkisini kullanarak risk seviyesini belirlemek

FL-5.2.2 (K2) Proje risklerinin ve ürün risklerinin farklarını belirtmek

FL-5.2.3 (K2) Ürün riski analizinin testlerin bütünlüğünü ve kapsamını nasıl etkileyebileceğini açıklamak

FL-5.2.4 (K2) Analiz edilen ürün risklerine karşı alınacak önlemleri açıklamak

5.3 Test Gözetimi, Test Kontrol ve Test Tamamlama

FL-5.3.1 (K1) Testlerde kullanılan metrikleri anımsamak

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

FL-5.3.3 (K2) Testin durumunun nasıl iletileceğine örnekler vermek

5.4 Yapılandırma Yönetimi

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

5.5 Hata Yönetimi

FL-5.5.1 (K3) Bir hata raporu hazırlamak

5.1. Test Planlama

5.1.1. Test Planının Amacı ve İçeriği

Test planı bir test projesi için hedefleri, kaynakları ve süreçleri belirtir. Test planı:

- Test hedeflerine ulaşmaya yarayan araçları ve programın dokümante edilmiş halidir
- Gerçekleştirilen test aktivitelerinin belirlenmiş kriterlerini karşılamasına yardımcı olur
- Ekip üyeleri ve diğer paydaşlarla iletişim için bir araç olarak hizmet eder
- Test sürecinin mevcut test politikasına ve test stratejisine uyacağını gösterir (veya testin bunlara neden uymayacağını açıklar)

Test planlama, test uzmanına düşünme konusunda rehberlik eder ve test uzmanını riskler, zaman çizelgeleri, insanlar, araçlar, masraflar, efor vb. ile ilgili konularda gelecekte ortaya çıkabilecek zorluklara karşı hazırlar. Test planı hazırlama süreci test projesi hedeflerine ulaşmak için gereken eforu hesaplamanın faydalı bir yoludur.

Tipik test planı içeriğinde şunlar yer alır:

- Test bağlamı (ör. kapsam, test hedefleri, kısıtlar, test esası)
- Test projesinin varsayımları ve kısıtları
- Paydaşlar (ör. roller, sorumluluklar, teste uygunluk, işe alım ve eğitim ihtiyaçları)
- İletişim (ör. iletişimin şekli ve sıklığı, dokümantasyon şablonları)
- Risk kaydı (ör. ürün riskleri, proje riskleri)
- Test yaklaşımı (ör. test seviyeleri, test çeşitleri, test teknikleri, test çıktıları, giriş kriterleri ve çıkış kriterleri, testin bağımsızlığı, toplanacak metrikler, test verisi gereksinimleri, test ortamı gereksinimleri, organizasyonel test politikasından sapmalar ve test stratejisi)
- Bütçe ve zaman çizelgesi

Test planı ve içeriği hakkında daha fazla bilgiye ISO/IEC/IEEE 29119-3 standardından ulaşılabilir.

5.1.2. Test Uzmanının Döngü ve Sürüm Planlamasına Katkısı

Döngüsel YGYD'lerinde tipik olarak iki tür planlama gerçekleşir: sürüm planlaması ve döngü planlaması.

Sürüm planlaması bir ürünün piyasaya sürülmesini planlar, ürün iş listesini tanımlar ve yeniden tanımlar ve daha kapsamlı kullanıcı hikayelerinin daha küçük kullanıcı hikayesi gruplarına dönüştürülmesini içerebilir. Ayrıca, tüm döngülerde test yaklaşımı ve test planının esası olarak da hizmet eder. Sürüm planlamasında yer alan test uzmanları, test edilebilir kullanıcı hikayelerinin ve kabul kriterlerinin yazım sürecine katılır (bkz. bölüm 4.5), proje ve kalite riski analizlerinde görev alır (bkz. bölüm 5.2), kullanıcı hikayeleriyle ilişkili test eforunu tahmin eder (bkz. bölüm 5.1.4), test yaklaşımını belirler ve sürüm için testi planlar.

Döngü planlaması, tek bir döngünün sonunu öngörmekte olup döngüye ait iş listesiyle ilgilidir. Döngü planlamasında yer alan test uzmanları, kullanıcı hikayelerine ilişkin detaylı bir risk analizine katılır, kullanıcı hikayelerinin test edilebilirliğini belirler, kullanıcı hikayelerini görevlere (özellikle test görevleri) ayırır, tüm test görevleri için test eforuna dair tahminde bulunur ve fonksiyonel ve fonksiyonel olmayan test nesnesi unsurlarını belirler ve iyileştirir.

5.1.3. Giriş Kriterleri ve Çıkış Kriterleri

Giriş kriterleri belirli bir aktivite için önkoşulları tanımlar. Giriş kriterleri karşılanmazsa faaliyetin daha zor olması, daha uzun sürmesi, daha maliyetli ve daha riskli olması muhtemeldir. Çıkış kriterleri bir aktivitenin tamamlandığını anlamak için nelere ulaşılması gerektiğini tanımlar. Test hedeflerine göre farklılık gösterebilecek olan giriş kriterleri ve çıkış kriterleri her test seviyesi için tanımlanmalıdır.

Tipik giriş kriterleri arasında şunlar yer alır: kaynak elverişliliği (ör. insanlar, araçlar, ortamlar, test verisi, bütçe, zaman), test yazılımı elverişliliği (ör. test esası, test edilebilir gereksinimler, kullanıcı hikayeleri, test senaryoları) ve test nesnesinin başlangıç kalite seviyesi (ör. tüm duman testleri geçmiştir).

Tipik çıkış kriterleri arasında şunlar yer alır: bütünlük ölçüleri (ör. ulaşılan kapsam seviyesi, çözülmemiş hata sayısı, hata yoğunluğu, başarısız test senaryosu sayısı) ve tamamlama kriterleri (ör. planlanan testler koşturuldu, statik test yapıldı, bulunan tüm hatalar raporlandı, tüm regresyon testleri otomatik hale getirildi).

Zamanın veya paranın bitmesi de geçerli bir çıkış kriteri olarak görülebilir. Diğer çıkış kriterleri yerine getirilmese bile paydaşlar daha fazla test yapmadan ürünü piyasaya sürmenin riskini gözden geçirip kabul ettiyse, bu şartlar altında testlere son verilmesi kabul edilebilir bir durumdur.

Çevik yazılım geliştirmede, çıkış kriterleri genellikle Tamamlandı Tanımı olarak adlandırılır ve ekibin piyasaya sürülebilir bir öğe için hedef metriklerini tanımlar. Bir kullanıcı hikayesinin geliştirme ve/veya test aktivitelerinin başlaması için yerine getirmesi gereken giriş kriterleri Hazır Tanımı olarak adlandırılır.

5.1.4. Tahminleme Teknikleri

Test eforu tahmini, bir test projesinin hedeflerine ulaşmak için gereken testle ilgili iş miktarını tahmin etmeyi içerir. Tahminin bir dizi varsayıma dayandığını ve her zaman tahmin hatasına tabi olduğunu paydaşlara açıkça belirtmek önemlidir. Küçük görevlere ilişkin yapılan tahminler genellikle büyük görev tahminlerinden daha doğrudur. Bu nedenle, büyük bir görev için tahminleme yapılırken görev daha küçük görevlere bölünerek bunlara yönelik tahminleme gerçekleştirilir.

Bu ders programında aşağıdaki dört tahminleme tekniği açıklanmaktadır.

Oranlara dayalı tahminleme. Bu metrik bazlı teknikte, kuruluş içinde önceden yapılmış projelerden edinilen rakamlar toplanır ve bu da benzer projeler için "standart" oranların elde edilmesini mümkün kılar. Bir kuruluşun kendi projelerinin oranları (ör. geçmiş verilerden elde edilen) genellikle tahminleme sürecinde kullanılacak en iyi kaynaktır. Bu standart oranlar yeni proje için test eforunun tahmin edilmesinde kullanılabilir. Örneğin, bir önceki projede "geliştirme-test" eforu oranı 3:2 ise ve mevcut projede geliştirme eforunun 600 kişi-gün olması bekleniyorsa test eforunun 400 kişi-gün olacağı tahmin edilebilir.

Dışdeğerleme. Bu metrik bazlı teknikte, verileri toplamak için mevcut projede mümkün olduğunca erken ölçümler yapılır. Yeterli gözlem yapıldıktan sonra geri kalan iş için gereken efor bu veriye dışdeğerleme uygulanarak (genelde matematiksel bir model ile) yaklaşık olarak belirlenebilir. Bu yöntem döngüsel YGYD'lerine çok uygundur. Örneğin, ekip, gelecek döngüdeki test eforunu son üç döngüden kaynaklı ortalama efor olarak tahmin edebilir.

Wideband Delphi. Bu döngüsel, uzmana dayalı teknikte uzmanlar tecrübeye dayalı tahminler yapar. Her uzman tek başına eforu tahmin eder. Sonuçlar toplanır ve üzerinde hemfikir olunan sınırların dışında olan sapmalar söz konusu ise uzmanlar mevcut tahminlerini tartışır. Her bir uzmanın yine tek başına olacak şekilde geri bildirime bağlı yeni bir tahminde bulunması istenir. Bu süreç bir fikir birliği sağlanana kadar tekrar edilir. Poker planlama tekniği, Çevik yazılım geliştirmede yaygın olarak kullanılan

Wideband Delphi varyantıdır. Poker planlama tekniğinde tahminler genelde efor boyutunu temsil eden sayılara sahip kartlar kullanılarak yapılır.

Üç noktalı tahminleme. Uzmana dayalı bu teknikte uzmanlar tarafından üç tahmin yapılır: En iyimser tahmin (a), en olası tahmin (m) ve en kötümser tahmin (b). Son tahmin (E) bunların ağırlıklı aritmetik ortalamasıdır. Bu tekniğin en sık kullanılan versiyonunda tahmin şu şekilde hesaplanır: E = (a + 4*m + b) / 6. Bu tekniğin avantajı uzmanların ölçüm hatasını hesaplayabilmesine izin vermesidir. SD = (b – a) / 6. Örneğin tahminler (kişi-saat cinsinden) a=6, m=9 ve b=18 olduğunda son tahmin 10±2 kişi-saattir (ör. 8 ve 12 kişi-saat arası) çünkü E = (6 + 4*9 + 18) / 6 = 10 ve SD = (18 – 6) / 6 = 2 şeklindedir.

Bunlar ve diğer birçok test tahminlemesi teknikleri için bkz. (Kan 2003, Koomen 2006, Westfall 2009).

5.1.5. Test Senaryosu Önceliklendirme

Test senaryoları ve test prosedürleri belirlenip test gruplarına ayrıldıktan sonra bu test grupları, koşum sırasını belirleyen bir test koşum çizelgesinde düzenlenebilir. Test senaryoları önceliklendirilirken farklı faktörler dikkate alınabilir. En yaygın kullanılan test senaryosu önceliklendirme stratejileri aşağıdaki gibidir:

- Test koşum sırasının risk analizi sonuçlarına dayalı olduğu riske dayalı önceliklendirme (bkz. bölüm 5.2.3). En önemli riskleri içeren test senaryosu ilk olarak yürütülür.
- Test koşum sırasının kapsama dayalı olduğu (ör. komut kapsama yüzdesi) önceliklendirme. En yüksek kapsama ulaşan test senaryoları ilk olarak yürütülür. Ek kapsam önceliklendirmesi denilen başka bir varyantta, en yüksek kapsama ulaşan test senaryosu ilk uygulanır; sonraki her bir test senaryosu en yüksek ek kapsama ulaşan test senaryosudur.
- Test koşum sırasının, baz alınan gereksinimlerin önceliklerine dayandığı gereksinime dayalı önceliklendirme. Gereksinim öncelikleri paydaşlar tarafından belirlenir. En önemli gereksinimlerle ilgili test senaryoları ilk olarak yürütülür.

İdeal olarak test senaryoları, yukarıdaki önceliklendirme stratejilerinden biri kullanılarak öncelik seviyelerine göre çalıştırılacak şekilde sıralanır. 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.

Test koşum sıralamasında kaynakların elverişliliği de dikkate alınmalıdır. Örneğin, sadece spesifik bir zaman aralığında mevcut olabilecek gerekli test araçları, test ortamları veya kişiler.

5.1.6. Test Piramidi

Test piramidi, farklı testlerin farklı ayrıntıları olabileceğini gösteren bir modeldir. Test piramidi modeli farklı hedeflerin farklı test otomasyonu seviyeleriyle desteklendiğini göstererek test otomasyonunda ve test eforu tahsisinde ekibi destekler. Piramit katmanları test gruplarını temsil eder. Katman ne kadar yüksekse test ayrıntısı, test izolasyonu ve test koşumu süresi o kadar düşük olur. Alt katmandaki testler küçük, izole ve hızlıdır ve küçük bir fonksiyonalite parçasını kontrol eder; dolayısıyla makul bir kapsama ulaşmak için genellikle çok sayıda test gerekir. Üst katman karmaşık, üst seviye ve uçtan uca testleri temsil eder. Bu üst seviye testler genelde alt katmanlardaki testlerden daha yavaştır ve genelde büyük bir fonksiyonalite parçasını kontrol eder; dolayısıyla makul bir kapsama ulaşmak için genellikle sadece birkaç test gerekir. Katmanların sayısı ve adlandırması değişebilir. Örneğin, orijinal test piramidi modeli (Cohn 2009) üç katmanı şu şekilde tanımlar: "birim testleri", "servis testleri" ve "UI testleri". Bir diğer popüler model ise

birim (bileşen) testlerini, entegrasyon (bileşen entegrasyonu) testleri ve uçtan uca testleri tanımlar. Diğer test seviyeleri (bkz. bölüm 2.2.1) de kullanılabilir.

5.1.7. Test Çeyrekleri

Brian Marick (Marick 2003, Crispin 2008) tarafından tanımlanan test çeyrekleri, test seviyelerini Çevik yazılım geliştirmede uygun test çeşitleri, aktiviteleri, test teknikleri ve çalışma ürünleri ile gruplandırır. Model, tüm uygun test çeşitlerinin ve test seviyelerinin YGYD'ne dahil edilmesini sağlamak için bunları görselleştirmede ve bazı test çeşitlerinin belirli test seviyeleriyle diğerlerinden daha alakalı olduğunu anlamada test yönetimini destekler. Bu model ayrıca geliştiriciler, test uzmanları ve iş birimleri dahil tüm paydaşlara test çeşitlerinin ayrımını yapmanın ve açıklamanın bir yolunu sunar.

Bu modelde testler iş odaklı veya teknoloji odaklı olabilir. Testler ayrıca ekibi destekleyebilir (geliştirmeye rehberlik edebilir) veya ürünü eleştirebilir (beklentilere karşı davranışını ölçebilir). Bu iki perspektifin kombinasyonu dört çeyreği belirler:

- Çeyrek Q1 (teknoloji odaklı ve ekibi desteklemek için). Bu çeyrekte bileşen ve bileşen entegrasyon testleri yer alır. Bu testler otomatikleştirilip CI sürecine dahil edilmelidir.
- Çeyrek Q2 (iş odaklı ve ekibi desteklemek için). Bu çeyrekte fonksiyonel testler, örnekler, kullanıcı hikayesi testleri, kullanıcı deneyimi prototipleri, API testi ve simülasyonlar yer alır. Bu testler kabul kriterlerini kontrol eder ve manuel ya da otomatik olabilir.
- Çeyrek Q3 (iş odaklı, ürünün kritiğini yapmak için). Bu çeyrekte keşif testi, kullanılabilirlik testi, kullanıcı kabul testi yer alır. Bu testler kullanıcı odaklıdır ve genellikle manueldir.
- Çeyrek Q4 (teknoloji odaklı, ürünün kritiğini yapmak için). Bu çeyrekte duman testleri ve fonksiyonel olmayan testler (kullanılabilirlik testi hariç) yer alır. Bu testler genellikle otomatiktir.

5.2. Risk Yönetimi

Organizasyonlar hedeflerine ulaşıp ulaşamayacaklarını ve bunun ne zaman gerçekleşeceğini belirsiz hale getiren birçok iç ve dış faktörle karşı karşıyadır (ISO 31000). Risk yönetimi organizasyonların hedeflere ulaşma olasılığını artırmasına, ürünlerinin kalitesini geliştirmesine ve paydaşların güvenini artırmasına olanak sağlar.

Temel risk yönetimi aktiviteleri:

- Risk analizi (risk belirleme ve risk değerlendirmesinden oluşur; bkz. bölüm 5.2.3)
- Risk kontrolü (risk azaltma ve risk gözetiminden oluşur; bkz. bölüm 5.2.4)

Test aktivitelerinin risk analizi ve risk kontrolüne göre seçildiği, önceliklendirildiği ve yönetildiği test yaklaşımına risk bazlı test denir.

5.2.1. Risk Tanımı ve Risk Özellikleri

Risk, sonucu olumsuz bir etkiye neden olacak potansiyel bir olay, tehlike, tehdit veya durumdur. Bir risk iki faktörle karakterize edilebilir:

- Risk olasılığı – riskin meydana gelme olasılığı (sıfırdan büyük ve birden küçük)
- Risk etkisi (zarar) – riskin meydana gelmesinin sonuçları

Bu iki faktör bir risk ölçüsü olan risk seviyesini belirtir. Risk seviyesi ne kadar yüksekse çözülmesi de o kadar önemlidir.

5.2.2. Proje Riskleri ve Ürün Riskleri

Yazılım testlerinde genelde iki tür riskle ilgilenilir: proje riskleri ve ürün riskleri.

Proje riskleri projenin yönetimi ve kontrolüyle ilgilidir. Proje risklerine örnek olarak:

- Organizasyonel sorunlar (ör. proje çıktılarındaki gecikme, hatalı tahmin, maliyet)
- İnsan kaynaklı sorunlar (ör. yetersiz beceri, çatışma, iletişim problemleri, personel eksiği)
- Teknik sorunlar (ör. proje kapsamının kontrolsüz artışı, eksik araç desteği)
- Tedarikçi sorunları (ör. tedarikçi teslimat başarısızlığı, tedarikçi iflası)

Proje riskleri ortaya çıktığında, projenin zaman çizelgesini, bütçesini veya kapsamını etkileyebilir; bu da projenin hedeflerine ulaşma kapasitesine etki eder.

Ürün riskleri ürünün gereksinimleriyle ilgilidir (ör. ISO 25010 kalite modelinde açıklanmıştır). Ürün riski örnekleri arasında şunlar yer alır: eksik ya da yanlış fonksiyonalite, hatalı hesaplamalar, çalışma zamanı hataları, zayıf mimari, etkin olmayan algoritmalar, yetersiz yanıt süresi, kötü kullanıcı deneyimi, güvenlik açıkları. Ürün riskleri, meydana geldiklerinde, aşağıdakiler de dahil olmak üzere çeşitli olumsuz sonuçlara yol açabilir:

- Kullanıcı memnuniyetsizliği
- Gelir, güven ve itibar kaybı
- Diğer paydaşlara zararlar
- Yüksek bakım maliyetleri, yardım masasının aşırı yoğunluğu
- Para cezaları
- Ekstrem durumlarda fiziksel zarar, yaralanma ve hatta ölüm

5.2.3. Ürün Riski Analizi

Test perspektifinden bakıldığında, ürün riski analizinin amacı, ürün riski konusunda farkındalık sağlayarak test eforunu, kalan ürün riskini en aza indirecek şekilde yönlendirmektir. İdeal olarak ürün riski analizi YGYD'nün erken aşamalarında başlar.

Ürün riski analizi risk belirleme ve risk değerlendirmesinden oluşur. Risk belirleme kapsamlı bir risk listesi oluşturulmasıyla ilgilidir. Paydaşlar riskleri çeşitli teknikler ve araçlar kullanarak belirleyebilir; ör. beyin fırtınası, çalıştaylar, görüşmeler veya neden-sonuç diyagramları. Risk değerlendirmesi şunları içerir: belirlenen risklerin kategorilere ayrılması, risk olasılığının, risk etkisinin ve seviyesinin belirlenmesi, önceliklendirme ve söz konusu riskleri ele almaya ilişkin yöntemler tasarlanması. Kategorilere ayırma işlemi risk azaltma aksiyonlarının atanmasına yardımcı olur çünkü genelde aynı kategoriye giren riskler benzer bir yaklaşımla hafifletilebilir.

Risk değerlendirmesinde nicel veya nitel bir yaklaşım ya da her ikisinin bir karması kullanılabilir. Nicel yaklaşımda risk seviyesi, risk olasılığı ile risk etkisinin çarpılmasıyla hesaplanır. Nitel yaklaşımda risk seviyesi risk matrisi kullanılarak belirlenebilir.

Ürün riski analizi, testlerin bütünlüğünü ve kapsamını etkileyebilir. Sonuçları aşağıdaki amaçlarla kullanılır:

- Gerçekleştirilecek testlerin kapsamını belirlemek
- Belirli test seviyelerini tespit etmek ve gerçekleştirilecek test çeşitlerini önermek
- Kullanılacak test tekniklerini ve ulaşılacak kapsamı belirlemek
- Her bir görev için gerekli olan test eforunu tahmin etmek
- Kritik hataların mümkün olduğunca erken bulunması için testleri önceliklendirmek
- Riski azaltmak için testlere ek olarak uygulanabilecek aktiviteleri belirlemek

5.2.4. Ürün Risk Kontrolü

Ürün risk kontrolü, belirlenen ve değerlendirilen ürün risklerine karşı alınan tüm önlemlerden oluşur. Ürün risk kontrolü risk azaltma ve risk gözetimini içerir. Risk azaltma, risk değerlendirmesinde önerilen aksiyonların uygulanarak risk seviyesinin azaltmasını içerir. Risk gözetiminin amacı risk azaltma aksiyonlarının etkili olmasını sağlamak, risk değerlendirmesini iyileştirmek için daha fazla bilgi elde etmek ve ortaya çıkan riskleri tanımlamaktır.

Ürün risk kontrolüne ilişkin olarak bir risk analiz edildiğinde riske yönelik birkaç yanıt seçeneği mümkündür; örneğin, test yoluyla riskin azaltılması, risk kabulü, risk transferi veya acil durum planı (Veenendaal 2012). Test yoluyla ürün riskini azaltmak için alınabilecek aksiyonlar aşağıdaki gibidir:

- Uygun deneyim ve beceri seviyesine sahip uygun test uzmanlarını seçmek
- Uygun bir test bağımsızlığı seviyesi uygulamak
- Gözden geçirmeler ve statik analizler gerçekleştirmek
- Uygun test teknikleri ve kapsam seviyeleri uygulamak
- Uygun test çeşitlerini uygulamak
- Regresyon testi de dahil olmak üzere dinamik test uygulamak

5.3. Test Gözetimi, Test Kontrol ve Test Tamamlama

Test gözetimi test hakkında bilgi toplamayla ilgilidir.
Bu bilgi, test ilerlemesini değerlendirmek ve test çıkış kriterlerinin veya çıkış kriterleriyle ilişkili test görevlerinin (ürün riskleri, gereksinimler veya kabul kriterlerinin kapsamına yönelik hedeflerin karşılanması gibi) karşılanıp karşılanmadığını ölçmek için kullanılır.

Test kontrolü, en etkili ve verimli testin gerçekleştirilmesi için kontrol yönergeleri şeklinde, rehberlik ve gerekli düzeltici aksiyonları sağlamak üzere test gözetiminden elde edilen bilgileri kullanır. Kontrol yönergelerine örnek olarak şunlar verilebilir:

- Belirlenen bir risk sorun haline geldiğinde testleri yeniden önceliklendirmek
- Bir test öğesinin giriş veya çıkış kriterlerine uyup uymadığını yeniden değerlendirmek
- Test ortamının teslimatındaki gecikmeleri ele almak için test zaman çizelgesini ayarlamak
- Gerektiği zaman ve yerde yeni kaynaklar eklemek

Test tamamlama, 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 test seviyesinin tamamlanması, bir çevik döngünün tamamlanması, bir test projesinin tamamlanması (veya iptali), bir yazılım sisteminin canlıya alınması veya bir bakım sürümünün tamamlanması gibi proje kilometre taşlarında gerçekleştirilir.

5.3.1. Yazılım Testlerinde Kullanılan Metrikler

Test metrikleri, planlanan programa ve bütçeye göre ilerlemeyi, test nesnesinin mevcut kalitesini ve test aktivitelerinin hedeflere veya döngüye göre etkinliğini göstermek için toplanır. Test gözetimi, test kontrol ve test tamamlama süreçlerinin desteklenmesi için çeşitli metrikler toplar.

Yaygın test metrikleri arasında şunlar yer alır:

- Proje ilerleme metrikleri (ör. görev tamamlama, kaynak kullanımı, test eforu)
- Test ilerleme metrikleri (ör. test senaryosu uyarlama ilerlemesi, test ortamı hazırlığı ilerlemesi, çalıştırılan/çalıştırılmayan, geçen/geçmeyen test senaryosu sayısı, test koşumu süresi)
- Ürün kalite metrikleri (ör. elverişlilik, yanıt süresi, arızlar arası ortalama süre)
- Hata metrikleri (ör. bulunan/düzeltilen hataların sayısı ve öncelikleri, hata yoğunluğu, hata tespit yüzdesi)
- Risk metrikleri (ör. kalan risk seviyesi)
- Kapsam metrikleri (ör. gereksinim kapsamı, kod kapsamı)
- Maliyet metrikleri (ör. test maliyeti, kalite maliyeti)

5.3.2. Test Raporlarının Amacı, İçeriği ve Hedef Kitlesi

Test raporunda test öncesi ve sonrasındaki test bilgileri özetlenir ve iletilir. Test ilerleme raporları testin devam eden kontrolünü destekler ve plandan sapan durumlar veya değişen durumlar nedeniyle değişikliğe ihtiyaç duyulduğunda test zaman çizelgesinde, kaynaklarda veya test planında değişiklik yapmaya olanak sağlayacak yeterli bilgiyi sağlamalıdır. Test tamamlama raporlarında testin belirli bir aşaması özetlenir (ör. test seviyesi, test döngüsü, yazılım döngüsü) ve sonraki testler için bilgi verilebilir.

Test gözetimi ve kontrolü sırasında, test ekibi paydaşları bilgilendirmek için test ilerleme raporları hazırlar. Test ilerleme raporu genelde düzenli olarak hazırlanır (ör. günlük, haftalık vb.) ve şunları içerir:

- Test periyodu
- Her tür önemli sapma da dahil olmak üzere test ilerleme durumu (ör. planın önünde veya gerisinde)
- Testi engelleyen durumlar ve bunların çözümü
- Test metrikleri (örnekler için bkz. bölüm 5.3.1)
- Test periyodunda ortaya çıkan yeni ve değişen riskler
- Sıradaki periyod için planlanan testler

Test tamamlama raporu testin tamamlanması sırasında, bir proje, test seviyesi veya test çeşidi tamamlandığında ve ideal olarak çıkış kriterleri karşılandığında hazırlanır. Bu rapor, test ilerleme raporu ve diğer verileri kullanır. Tipik test tamamlama raporlarında şunlar yer alır:

- Test özeti
- Orijinal test planına dayalı test ve ürün kalitesi değerlendirmesi (ör. test hedefleri ve çıkış kriterleri)
- Test planından sapmalar (ör. planlanan zaman çizelgesine, süreye ve efora göre yaşanan farklılıklar).
- Test engelleri ve çözümler
- Test ilerleme raporuna dayalı test metrikleri
- Etkisi azaltılmamış riskler, düzeltilmemiş hatalar
- Testle ilgili olarak alınan dersler

Farklı hedef kitleleri için raporlarda farklı bilgiler gereklidir ve farklı hedef kitleleri raporların resmiyet derecesini ve sıklığını etkiler. Aynı ekipteki diğer kişilere test ilerleme durumu hakkında rapor verilmesi genellikle sık ve gayri resmi iken, tamamlanmış bir proje için test raporlaması belirli bir şablonu izler ve yalnızca bir kez gerçekleşir.

ISO/IEC/IEEE 29119-3 standardı test ilerleme raporu (test durum raporu da denir) ve test tamamlama raporu için şablon ve örnekler içerir.

5.3.3. Testin Durumunun Bildirilmesi

Test durumunu bildirmenin en iyi yolu, test yönetimi endişelerine, organizasyonel test stratejilerine, düzenleyici standartlara veya kendi kendini yöneten ekipler söz konusu olduğunda (bkz. bölüm 1.5.2) ekibin kendisine bağlı olarak değişir. Seçenekler arasında şunlar yer alır:

- Ekip üyeleri ve diğer paydaşlarla sözlü iletişim
- Gösterge panelleri (ör. CI/CD gösterge panelleri, görev panoları ve burn-down tabloları)
- Elektronik iletişim kanalları (ör. e-posta, sohbet)
- Çevrim içi dokümantasyon
- Resmi test raporları (bkz. bölüm 5.3.2)

Bu seçeneklerden bir veya daha fazlası kullanılabilir. Coğrafi mesafe veya zaman farklılıkları nedeniyle doğrudan yüz yüze iletişimin her zaman mümkün olmadığı dağınık ekipler için daha resmi bir iletişim daha uygun olabilir. Genelde farklı paydaşlar farklı bilgi çeşitleriyle ilgilenir, dolayısıyla iletişim buna göre ayarlanmalıdır.

5.4. Yapılandırma Yönetimi

Test sürecinde yapılandırma yönetimi (YY - Configuration Management) test planı, test stratejisi, test koşulları, test senaryosu, test betiği, test sonucu, test kaydı ve test raporu gibi çalışma ürünlerini yapılandırma öğeleri olarak tanımlamak, kontrol etmek ve takip etmek için bir disiplin sağlar.

Karmaşık bir yapılandırma öğesi için (ör. test ortamı) YY içerdiği öğeleri, aralarındaki ilişkileri ve versiyonlarını kaydeder. Yapılandırma öğesi test için onaylandığında temel çizgi (baseline) haline gelir ve sadece resmi bir değişiklik kontrol süreci aracılığıyla değiştirilebilir.

Yeni bir temel çizgi oluşturulduğunda, yapılandırma yönetimi değiştirilen yapılandırma öğelerinin bir kaydını tutar. Önceki test sonuçlarını yeniden üretmek için önceki temel çizgiye geri dönülebilir.

Test sürecinin uygun şekilde desteklenmesi için YY aşağıdakilerin gerçekleştirilmesini sağlar:

- Test öğeleri de dahil olmak üzere tüm yapılandırma öğelerinin benzersiz olarak tanımlanması, versiyon kontrolü yapılmış, değişiklikleri izlenmiş ve diğer yapılandırma öğeleriyle ilgili olması ve böylece test süreci boyunca izlenebilirliğinin sürdürülmesi
- Tanımlanan tüm dokümantasyon ve yazılım öğelerinin test dokümantasyonunda açık bir şekilde belirtilmesi

Sürekli entegrasyon, sürekli teslimat, sürekli dağıtım ve ilgili testler genellikle otomatikleştirilmiş DevOps hattının (bkz. bölüm 2.1.4) bir parçası olarak uygulanır ve otomatikleştirilmiş YY de normalde buna dahildir.

5.5. Hata Yönetimi

Önemli test hedeflerinden biri hataları bulmak olduğundan yerleşik bir hata yönetimi süreci olması çok önemlidir. Burada "hatalar"dan bahsediyor olsak da, raporlanan anomalilerin gerçek hatalar veya başka bir şey (ör. yanlış pozitif, değişiklik talebi) olduğu ortaya çıkabilir. Bu, hata raporlarıyla ilgilenme sürecinde çözülür. Anomaliler YGYD'nün herhangi bir aşamasında raporlanabilir ve raporlama şekli YGYD'ne bağlıdır. Hata yönetimi süreci asgari olarak, anomalilerin keşfedilmesinden kapatılmasına kadar tek tek ele alınmasına yönelik bir iş akışını ve sınıflandırılmalarına ilişkin kuralları içerir. İş akışı tipik olarak raporlanan anomalileri kaydetme, analiz etme ve sınıflandırma, düzeltme veya olduğu şekliyle tutma gibi uygun bir yanıta karar verme ve son olarak hata raporunu kapatma faaliyetlerini içerir. Süreç, işe dahil olan tüm paydaşlar tarafından izlenmelidir. Statik testlerde (özellikle statik analiz) bulunan hataların da benzer şekilde ele alınması tavsiye edilir.

Hata raporlarının hedefleri genellikle aşağıdaki gibidir:

- Raporlanan hataları ele almaktan ve çözmekten sorumlu olanlara sorunu çözmek için yeterli bilgiyi sağlamak
- Çalışma ürününün kalitesini izlemek için bir araç sağlamak
- Geliştirme ve test sürecinin iyileştirilmesi için fikirler sunmak

Dinamik test sırasında kaydedilen hata raporu tipik olarak şunları içerir:

- Benzersiz tanımlayıcı
- Raporlanan anomalinin başlığı ve kısa bir özeti
- Anomalinin gözlemlendiği tarih, bildiren organizasyon ve yazar (rolü de dahil olmak üzere)
- Test nesnesinin ve test ortamının tanımı
- Hatanın bağlamı (ör. çalıştırılan test senaryosu, yapılan test aktivitesi, YGYD aşaması ve test tekniği, kontrol listesi ve kullanılan test verisi gibi diğer ilgili bilgiler)

- Anomaliyi tespit eden adımlar da dahil olmak üzere anomalinin yeniden oluşturulmasını ve çözümünü sağlamak üzere arızanın açıklaması, ilgili test kayıtları, veritabanı dökümleri, ekran görüntüleri veya kayıtlar
- Beklenen sonuçlar ve gerçekleşen sonuçlar
- Hatanın paydaşların çıkarları veya gereksinimler üzerindeki önem derecesi (etki derecesi)
- Düzeltilme önceliği
- Hatanın durumu (ör. açık, ertelenmiş, tekrarlanmış, düzeltilmeyi bekliyor, onaylama testi bekleniyor, yeniden açıldı, kapatıldı, reddedildi)
- Referanslar (ör. test senaryosuna ilişkin)

Bu verilerin bir kısmı hata yönetim araçları kullanıldığında otomatik olarak dahil edilebilir (ör. tanımlayıcı, tarih, yazar ve başlangıç durumu). Hata raporu için doküman şablonları ve örnek hata raporları, hata raporlarını olay raporları olarak açıklayan ISO/IEC/IEEE 29119-3 standardında bulunabilir.

6. Test Araçları – 20 dakika

Anahtar kelimeler

test otomasyonu

Konu 6 Öğrenme Hedefleri:

6.1 Yazılım Testleri için Araç Desteği

FL-6.1.1 (K2) Farklı test araçlarının test sürecini nasıl desteklediğini açıklamak

6.2 Test Otomasyonunun Faydaları ve Riskleri

FL-6.2.1 (K1) Test otomasyonunun faydalarını ve risklerini hatırlamak

6.1. Yazılım Testleri için Araç Desteği

Test araçları birçok test aktivitesini destekler ve kolaylaştırır. Örnekler aşağıdakileri içerir ancak bunlarla sınırlı değildir:

- Yönetim araçları: YGYD yönetimini, gereksinimleri, testleri, hataları ve yapılandırmayı kolaylaştırarak test sürecinin verimliliğini artırır
- Statik test araçları: Gözden geçirmeler ve statik analiz konusunda test uzmanına destek sağlar
- Test tasarımı ve uygulama araçları: Test senaryoları, test verisi ve test prosedürlerini oluşturmayı kolaylaştırır
- Test koşumu ve kapsam araçları: Otomatik test koşumu ve kapsam ölçümünü kolaylaştırır
- Fonksiyonel olmayan test araçları: Test uzmanının manuel olarak yapılması zor veya imkansız olan fonksiyonel olmayan testleri yapmasına olanak sağlar
- DevOps araçları: DevOps teslimat hattını, iş akışı takibini, otomatik derleme süreçlerini, CI/CD'yi destekler
- İş birliği araçları: İletişimi kolaylaştırır
- Ölçeklenebilirliği ve dağıtım standardizasyonunu destekleyen araçlar (ör. sanal makineler, konteynerleştirme araçları)
- Teste yardımcı her türlü araç (ör. elektronik tablolar, test etme bağlamında kullanılan bir test aracıdır)

6.2. Test Otomasyonunun Faydaları ve Riskleri

Sadece test aracına odaklanıp bunu hayata geçirmeye çalışmak iyi bir test pratiğinin garantisi değildir. Her yeni araç, gerçek ve kalıcı faydalar sağlamak için efor gerektirmektedir (ör. aracın kullanıma alınması, bakım ve eğitim). Analiz edilmesi ve hafifletilmesi gereken bazı riskler de vardır.

Test otomasyonu kullanmanın potansiyel faydaları şunlardır:

- Tekrarlayan manuel işleri azaltarak zaman tasarrufu sağlaması (ör. regresyon testleri yürütme, aynı test verilerini yeniden girme, beklenen sonuçlarla gerçekleşen sonuçları karşılaştırma ve kodlama standartlarına göre kontrol etme)
- Daha fazla tutarlılık ve tekrarlanabilirlik sayesinde basit insan hatalarını önlemesi (ör. gereksinimlerden tutarlı bir şekilde testler türetilmesi, test verilerinin sistematik bir şekilde oluşturulması ve testlerin bir araç tarafından aynı sırada ve aynı sıklıkta yürütülmesi)
- Daha objektif değerlendirme (ör. kapsam) ve insanlar tarafından bulunması çok karmaşık ölçüler sağlaması
- Test yönetimi ve test raporunu desteklemek için test hakkında gereken bilgilere daha kolay erişim (ör. istatistikler, grafikler ve test ilerleme durumu, hata oranları ve test koşumu süresi hakkında toplanmış veriler)
- Test koşum süresinin kısalmasıyla daha erken hata tespiti, daha hızlı geri bildirim ve daha hızlı pazara sürüm süresi

- Yeni, daha derin ve daha etkin testler tasarlamaları için test uzmanlarına daha fazla süre sağlaması

Test otomasyonu kullanmanın olası riskleri şunlardır:

- Bir aracın faydaları konusunda gerçekçi olmayan beklentiler (fonksiyonalite ve kullanım kolaylığı gibi).
- Bir aracı kullanıma almak, test betiklerini sürdürmek ve mevcut manuel test sürecini değiştirmek için gereken zaman, maliyet ve efora ilişkin yanlış tahminler.
- Manuel testlerin daha uygun olduğu durumlarda bir test aracı kullanmak.
- Kullanılan otomasyon aracıyla ilgili beklentinin yüksek olması (Ör. İnsanlar tarafından yapılması gereken eleştirel düşünmeyi yok saymak)

Faaliyetlerine son verme, aracı kullanımdan kaldırma, aracı farklı bir satıcıya satma veya yetersiz destek sağlama ihtimali olan bir araç satıcısına bağımlı kalmak (ör. sorulara yanıt vermeme,araç için güncelleme gelmemesi veya geç gelmesi, araçta karşılaşılan hata düzeltmelerinin yapılmaması veya geç yapılması gibi).

- Artık kullanılmayan, diğer bir deyişle artık güncelleme yapılmayan veya dahili bileşenleri için geliştirme amaçlı çok sık güncelleme gerekebilecek açık kaynaklı yazılımlar kullanmak.
- Otomasyon aracının geliştirme platformuyla uyumlu olmaması.
- Düzenleyici gereksinimlere ve/veya emniyet standartlarına uygun olmayan bir araç seçmek.

7. Referanslar

Standartlar

ISO/IEC/IEEE 29119-1 (2022) Yazılım ve sistem mühendisliği - Yazılım testleri - Bölüm 1: Genel Konseptler

ISO/IEC/IEEE 29119-2 (2021) Yazılım ve sistem mühendisliği - Yazılım testleri - Bölüm 2: Test süreçleri

ISO/IEC/IEEE 29119-3 (2021) Yazılım ve sistem mühendisliği - Yazılım testleri - Bölüm 3: Test dokümantasyonu

ISO/IEC/IEEE 29119-4 (2021) 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

ISO/IEC/IEEE 14764:2022 - Yazılım mühendisliği - Yazılım yaşam döngüsü süreçleri - Bakım ISO 31000

(2018) Risk yönetimi - Prensipler ve yönergeler

Kitaplar

Adzic, G. (2009) Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing, Neuri Limited

Ammann, P. ve Offutt, J. (2016) Introduction to Software Testing (2e), Cambridge University Press

Andrews, M. ve Whittaker, J. (2006) How to Break Web Software: Functional and Security Testing of Web Applications and Web Services, Addison-Wesley Professional

Beck, K. (2003) Test Driven Development: By Example, Addison-Wesley

Beizer, B. (1990) Software Testing Techniques (2e), Van Nostrand Reinhold: Boston MA Boehm, B. (1981)

Software Engineering Economics, Prentice Hall, Englewood Cliffs, NJ

Buxton, J.N. and Randell B., eds (1970), Software Engineering Techniques. NATO Bilim Komitesi, Roma, İtalya tarafından desteklenen bir konferansa ilişkin rapor, 27–31 Ekim 1969, sf. 16

Chelimsky, D. ve ark. (2010) The Rspec Book: Behaviour Driven Development with Rspec, Cucumber, and Friends, The Pragmatic Bookshelf: Raleigh, NC

Cohn, M. (2009) Succeeding with Agile: Software Development Using Scrum, Addison-Wesley Copeland,

L. (2004) A Practitioner's Guide to Software Test Design, Artech House: Norwood MA Craig, R. ve Jaskiel,

S. (2002) Systematic Software Testing, Artech House: Norwood MA

Crispin, L. ve Gregory, J. (2008) Agile Testing: A Practical Guide for Testers and Agile Teams, Pearson Education: Boston MA

Forgács, I. ve Kovács, A. (2019) Practical Test Design: Selection of traditional and automated test design techniques, BCS, The Chartered Institute for IT

Gawande A. (2009) The Checklist Manifesto: How to Get Things Right, New York, NY: Metropolitan Books

Gärtner, M. (2011), ATDD by Example: A Practical Guide to Acceptance Test-Driven Development, Pearson Education: Boston MA

Gilb, T., Graham, D. (1993) Software Inspection, Addison Wesley

Hendrickson, E. (2013) Explore It!: Reduce Risk and Increase Confidence with Exploratory Testing, The Pragmatic Programmers

Hetzel, B. (1988) The Complete Guide to Software Testing, 2. baskı, John Wiley ve Sons

Jeffries, R., Anderson, A., Hendrickson, C. (2000) Extreme Programming Installed, Addison-Wesley Professional

Jorgensen, P. (2014) Software Testing, A Craftsman's Approach (4e), CRC Press: Boca Raton FL

Kan, S. (2003) Metrics and Models in Software Quality Engineering, 2. baskı, Addison-Wesley

Kaner, C., Falk, J. ve Nguyen, H.Q. (1999) Testing Computer Software, 2. baskı, Wiley

Kaner, C., Bach, J. ve Pettichord, B. (2011) Lessons Learned in Software Testing: A Context-Driven Approach, 1. baskı, Wiley

Kim, G., Humble, J., Debois, P. ve Willis, J. (2016) The DevOps Handbook, Portland, OR

Koomen, T., van der Aalst, L., Broekman, B. ve Vroon, M. (2006) TMap Next for result-driven testing, UTN Publishers, The Netherlands

Myers, G. (2011) The Art of Software Testing, (3e), John Wiley & Sons: New York NY

O'Regan, G. (2019) Concise Guide to Software Testing, Springer Nature Switzerland

Pressman, R.S. (2019) Software Engineering. A Practitioner's Approach, 9. baskı, McGraw Hill

Roman, A. (2018) Thinking-Driven Testing. The Most Reasonable Approach to Quality Control, Springer Nature Switzerland

Van Veenendaal, E (ed.) (2012) Practical Risk-Based Testing, The PRISMA Approach, UTN Publishers: The Netherlands

Watson, A.H., Wallace, D.R. ve McCabe, T.J. (1996) Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, U.S. Dept. of Commerce, Technology Administration, NIST

Westfall, L. (2009) The Certified Software Quality Engineer Handbook, ASQ Quality Press

Whittaker, J. (2002) How to Break Web Software: A Practical Guide to Testing, Pearson

Whittaker, J. (2009) Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design, Addison Wesley

Whittaker, J. ve Thompson, H. (2003) How to Break Software Security, Addison Wesley

Wiegers, K. (2001) Peer Reviews in Software: A Practical Guide, Addison-Wesley Professional

Makaleler ve Web Sayfaları

Brykczynski, B. (1999) "A survey of software inspection checklists," *ACM SIGSOFT Software Engineering Notes, 24(1), sf. 82-89*

Enders, A. (1975) "An Analysis of Errors and Their Causes in System Programs," *IEEE Transactions on Software Engineering 1(2), sf. 140-149*

Manna, Z., Waldinger, R. (1978) "The logic of computer programming," *IEEE Transactions on Software Engineering 4(3), sf. 199-229*

Marick, B. (2003) Exploration through Example, http://www.exampler.com/old- blog/2003/08/21.1.html#agile-testing-project-1

Nielsen, J. (1994) "Enhancing the explanatory power of usability heuristics," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems: Celebrating Interdependence, ACM Press, sf. 152–158*

Salman. I. (2016) "Cognitive biases in software quality and testing," *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16), ACM, sf. 823-826.*

Wake, B. (2003) "INVEST in Good Stories, and SMART Tasks," https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

8. Ek A - Öğrenme Hedefleri/Bilginin Bilişsel Seviyesi

Bu ders programı aşağıdaki öğrenme hedeflerini içerir. Ders programındaki her konu, ilgili öğrenme hedefine göre incelenecektir. Öğrenme hedefleri aşağıda belirtildiği üzere bilişsel bilgi seviyesine denk gelen bir aksiyon fiiliyle başlar.

Seviye 1: Hatırla (K1): Aday, terim ve kavramları tanımalı ve hatırlamalıdır.

Aksiyon fiilleri: tanımlama, geri çağırma, hatırlama, tanıma.

Örnekler:

- "Genel test hedeflerini tanımlamak"
- "Test piramidinin konseptlerini hatırlamak"
- "Bir test uzmanının döngü ve sürüm planlamasına nasıl değer kattığını anlamak"

Seviye 2: Anla (K2): Aday, konuyla ilgili ifadelerin nedenlerini veya açıklamalarını seçebilir ve test konsepti için özetleme, karşılaştırma, sınıflandırma yapabilir ve örnekler verebilir.

Aksiyon fiilleri: sınıflandırma, karşılaştırma, mukayese, ayırt etme, örneklendirme, açıklama, örnek verme, yorumlama, özetleme.

Örnekler:

- "Yazım kabul kriterleri için farklı seçenekleri sınıflandırmak"
- "Test etme sürecindeki farklı rolleri karşılaştırmak" (benzerlikleri, farklılıkları veya her ikisini aramak)
- "Proje risklerinin ve ürün risklerinin farklarını belirtmek" (konseptlerin ayırt edilmesini sağlar).
- "Bir test planının amacına ve içeriğine ilişkin örnekler vermek"
- "Proje bağlamının test süreci üzerindeki etkisini açıklamak"
- "Gözden geçirme süreci faaliyetlerini özetlemek"

Seviye 3: Uygula (K3): Aday, aşina olduğu bir görevle karşılaştığında bir prosedürü gerçekleştirebilir veya doğru prosedürü seçip bunu belirli bir bağlam dahilinde uygulayabilir.

Aksiyon fiilleri: uygulama, hazırlama, kullanma.

Örnekler:

- Test senaryosu önceliklendirme işlemi uygulamak" (bir prosedür, teknik, süreç, algoritmayı vb. işaret etmelidir).
- "Bir hata raporu hazırlamak"
- "Test senaryoları elde etmek için sınır değer analizi kullanmak"

Öğrenme hedeflerinin bilişsel seviyeleri ile ilgili **referanslar**:

Anderson, L. W. ve Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching

Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn & Bacon

9. Ek B – Öğrenme Hedefleriyle İş Çıktıları izlenebilirlik matrisi

Bu bölüm, İş Çıktılarıyla ilgili Temel Seviye Öğrenme Hedeflerinin sayısını ve Temel Seviye İş Çıktıları ile Temel Seviye Öğrenme Hedefleri arasındaki izlenebilirliği belirtir.

İş Çıktıları: Temel Seviye		FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
BO1	Test etmenin ne olduğunu ve neden yararlı olduğunu anlamak	6													
BO2	Yazılım testinin temel konseptlerini anlamak		22												
BO3	Testin bağlamına göre uygulanacak yaklaşımı ve aktiviteleri tanımlamak			6											
BO4	Dokümantasyon kalitesini değerlendirmek ve iyileştirmek				9										
BO5	Testin etkinliğini ve verimliliğini artırmak					20									
BO6	Test sürecini yazılım geliştirme yaşam döngüsüyle aynı paralele getirmek						6								
BO7	Test yönetimi prensiplerini anlamak							6							
BO8	Net ve anlaşılır hata raporları yazmak ve bunları aktarmak								1						
BO9	Test etmeyle ilgili öncelik ve eforu etkileyen faktörleri anlamak									7					
BO10	Görevler arası bir ekibin üyesi olarak çalışabilir										8				
BO11	Test otomasyonuyla ilgili riskleri ve yararları bilmek											1			
BO12	Test etme sürecinde gerekli olan becerileri tanımlamak												5		
BO13	Riskin test üzerindeki etkisini anlamak													4	
BO14	Testin ilerleme durumu ve test kalitesi hakkında etkin bir rapor sunmak														4

Konu / bölüm / alt bölüm	Öğrenme hedefi	K-seviyesi	İŞ ÇIKTILARI														
			FL-BO1	FL-BO2	FL-BO3	FL-BO4	FL-BO5	FL-BO6	FL-BO7	FL-BO8	FL-BO9	FL-BO10	FL-BO11	FL-BO12	FL-BO13	FL-BO14
Konu 1	**Yazılım Testinin Temelleri**															
1.1	**Yazılım Testi nedir?**															
1.1.1	Genel test hedeflerini tanımlamak	K1	X													
1.1.2	Test ile hata ayıklamanın farklarını belirtmek	K2		X												
1.2	**Yazılım Testi Neden Gereklidir?**															
1.2.1	Örnekler vererek yazılım testinin neden gerekli olduğunu açıklamak	K2	X													
1.2.2	Yazılım testi ve kalite güvence arasındaki ilişkiyi anımsamak	K1		X												
1.2.3	Kök neden, insan hatası, hata ve arıza arasındaki farkı ayırt etmek	K2		X												
1.3	**Test Prensipleri**															
1.3.1	Yedi test prensibini açıklamak	K2		X												
1.4	**Test Aktiviteleri, Test Yazılımı ve Test Rolleri**															
1.4.1	Farklı test aktivitelerini ve görevlerini özetlemek	K2			X											
1.4.2	Proje bağlamının test süreci üzerindeki etkisini açıklamak	K2			X			X								
1.4.3	Test aktivitelerini destekleyen test yazılımlarını ayırt etmek	K2			X											
1.4.4	İzlenebilirliğin sağlanmasının önemini açıklamak	K2				X	X									
1.4.5	Test etme sürecindeki farklı rolleri karşılaştırmak	K2										X				

1.5	**Test Etme Sürecinde Gerekli Beceriler ve İyi Uygulamalar**															
1.5.1	Test etme sürecinde gerekli genel becerilere örnekler vermek	K2												X		
1.5.2	Tüm ekip yaklaşımının avantajlarını hatırlamak	K1										X				
1.5.3	Test bağımsızlığının yararlarını ve sakıncalarını açıklamak	K2			X											
Konu 2	**Yazılım Geliştirme Yaşam Döngüsü Boyunca Test**															
2.1	**Yazılım Geliştirme Yaşam Döngüsü Bağlamında Test**															
2.1.1	Seçilen yazılım geliştirme yaşam döngüsünün test üzerindeki etkisini açıklamak	K2						X								
2.1.2	Tüm yazılım geliştirme yaşam döngüsü için geçerli olan iyi test uygulamalarını hatırlamak	K1						X								
2.1.3	Geliştirmeye yönelik önce-test-et yaklaşımlarına örnekler vermek	K1					X									
2.1.4	DevOps'ların test üzerinde nasıl bir etkisi olabileceğini özetlemek	K2					X	X			X	X				
2.1.5	Shift-left yaklaşımını açıklamak	K2					X	X								
2.1.6	Geçmişe dönük verilerin süreç iyileştirmesi için nasıl bir mekanizma olarak kullanılabileceğini açıklamak	K2					X					X				
2.2	**Test Seviyeleri ve Test Çeşitleri**															
2.2.1	Farklı test seviyelerini ayırt etmek	K2		X	X											
2.2.2	Farklı test çeşitlerini ayırt etmek	K2		X												
2.2.3	Onaylama testleri ve regresyon testlerini birbirinden ayırmak	K2		X												
2.3	**Bakım Testleri**															
2.3.1	Bakım testleri ve tetikleyicilerini özetlemek	K2		X					X							
Konu 3	**Statik Testler**															
3.1	**Statik Testin Temelleri**															

3.1.1	Farklı statik test teknikleri ile incelenebilecek ürün çeşitlerini ayırt etmek	K1				X	X										
3.1.2	Statik testin önemini açıklamak	K2	X			X	X										
3.1.3	Statik ve dinamik testleri karşılaştırmak	K2				X	X										
3.2	**Geri Bildirim ve Gözden Geçirme Süreci**																
3.2.1	Erken ve sık paydaş geri bildiriminin faydalarını tanımlamak	K1	X			X						X					
3.2.2	Gözden geçirme süreci faaliyetlerini özetlemek	K2			X	X											
3.2.3	Gözden geçirme yapılırken ana rollere atanan sorumlulukları hatırlamak	K1				X								X			
3.2.4	Farklı gözden geçirme çeşitlerini karşılaştırmak	K2		X													
3.2.5	Başarılı bir gözden geçirmeye katkıda bulunan faktörleri hatırlamak	K1					X							X			
Konu 4	**Test Analizi ve Tasarımı**																
4.1	**Test Tekniklerine Genel Bakış**																
4.1.1	Kara kutu, beyaz kutu ve tecrübeye dayalı test tekniklerini ayırmak	K2		X													
4.2	**Kara Kutu Test Teknikleri**																
4.2.1	Test senaryoları elde etmek için denklik paylarına ayırma işlemini kullanmak	K3					X										
4.2.2	Test senaryoları elde etmek için sınır değer analizi kullanmak	K3					X										
4.2.3	Test senaryoları elde etmek için karar tablosu testi kullanmak	K3					X										
4.2.4	Test senaryoları elde etmek için durum geçişi testi kullanmak	K3					X										
4.3	**Beyaz Kutu Test Teknikleri**																
4.3.1	Komut testini açıklamak	K2		X													

4.3.2	Dal testini açıklamak	K2		X												
4.3.3	Beyaz kutu testinin önemini açıklamak	K2	X	X												
4.4	**Tecrübeye Dayalı Test Teknikleri**															
4.4.1	Hata tahminlemeyi açıklamak	K2		X												
4.4.2	Keşif testini açıklamak	K2		X												
4.4.3	Kontrol listesine dayalı testi açıklamak	K2		X												
4.5	**İş Birliğine Dayalı Test Yaklaşımları**															
4.5.1	Geliştiriciler ve iş temsilcileri ile iş birliği içinde nasıl kullanıcı hikayeleri yazılacağını açıklamak	K2				X						X				
4.5.2	Yazım kabul kriterleri için farklı seçenekleri sınıflandırmak	K2										X				
4.5.3	Test senaryoları elde etmek için kabul testi güdümlü yazılım geliştirmeyi (ATDD) kullanmak	K3					X									
Konu 5	**Test Aktivitelerini Yönetme**															
5.1	**Test Planlama**															
5.1.1	Bir test planının amacına ve içeriğine ilişkin örnekler vermek	K2		X					X							
5.1.2	Bir test uzmanının döngü ve sürüm planlamasına nasıl değer kattığını anlamak	K1	X									X		X		
5.1.3	Giriş kriterleri ve çıkış kriterlerini karşılaştırıp kıyaslamak	K2				X		X								X
5.1.4	Gerekli test eforunu hesaplamak için tahminleme teknikleri kullanmak	K3							X		X					
5.1.5	Test senaryosu önceliklendirme işlemi uygulamak	K3							X		X					
5.1.6	Test piramidinin konseptlerini hatırlamak	K1		X												
5.1.7	Test çeyreklerini ve bunların test seviyeleri ve test çeşitleri ile olan ilişkilerini özetlemek	K2		X							X					

5.2	**Risk Yönetimi**															

5.2.1	Risk olasılığı ve risk etkisini kullanarak risk seviyesini belirlemek	K1							X						X	
5.2.2	Proje risklerinin ve ürün risklerinin farklarını belirtmek	K2		X											X	
5.2.3	Ürün riski analizinin testlerin bütünlüğünü ve kapsamını nasıl etkileyebileceğini açıklamak	K2					X				X				X	
5.2.4	Analiz edilen ürün risklerine karşı alınacak önlemleri açıklamak	K2		X			X								X	
5.3	**Test Gözetimi, Test Kontrol ve Test Tamamlama**															
5.3.1	Testlerde kullanılan metrikleri anımsamak	K1									X					X
5.3.2	Test raporlarının amaçlarını, içeriklerini ve hedef kitlelerini özetlemek	K2					X				X					X
5.3.3	Test durumunun nasıl iletileceğine örnekler vermek	K2												X		X
5.4	**Yapılandırma Yönetimi**															
5.4.1	Yapılandırma yönetiminin testleri nasıl desteklediğini özetlemek	K2					X		X							
5.5	**Hata Yönetimi**															
5.5.1	Bir hata raporu hazırlamak	K3		X						X						
Konu 6	**Test Araçları**															
6.1	**Yazılım Testleri için Araç Desteği**															
6.1.1	Farklı test araçlarının test sürecini nasıl desteklediğini açıklamak	K2					X									
6.2	**Test Otomasyonunun Faydaları ve Riskleri**															
6.2.1	Test otomasyonunun faydalarını ve risklerini hatırlamak	K1					X						X			

10. Ek C – Sürüm Notları

ISTQB® Temel Seviye Ders Programı v4.0, Temel Seviye ders programı (v3.1.1) ve Çevik Test Uzmanı 2014 ders programının büyük ölçüde güncellenmiş halidir. Bu nedenle, konu ve bölüme göre ayrıntılı sürüm notları yoktur. Ancak, ana değişikliklerin bir özeti aşağıda verilmiştir. Ek olarak, ayrı bir Sürüm Notları dokümanında ISTQB® , Temel Seviye Ders Programının 3.1.1 versiyonundaki ve Çevik Test Uzmanı Ders Programının 2014 versiyonundaki öğrenme hedefleri (ÖH) ile yeni Temel Seviye v4.0 Ders Programının öğrenme hedefleri arasında, hangilerinin eklendiğini, güncellendiğini veya çıkarıldığını göstererek izlenebilirlik sağlamıştır.

Ders programının yazıldığı sırada (2022-2023) 100'den fazla ülkede bir milyondan fazla kişi Temel Seviye sınavına girmiş ve dünya çapında 800.000'den fazla kişi 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 topluluğuna sunduğu kalite düzeyi hakkında yüz binlerce kişinin görüşlerini geliştirmek için yapılmıştır.

Bu versiyonda tüm ÖH'ler en küçük parçasına ayrılacak ve ÖH'ler ile ders programı bölümleri arasında bire bir izlenebilirlik sağlayacak şekilde düzenlenmiştir, dolayısıyla ÖH ile ilişkilendirilmemiş bir içerik yoktur. Amaç, pratik kullanım ve bilgi ve beceriler arasındaki dengeyi artırmaya odaklanarak bu versiyonun okunmasını, anlaşılmasını, öğrenilmesini ve tercüme edilmesini kolaylaştırmaktır.

Bu ana sürüm aşağıdaki değişiklikleri içermektedir:

- Ders programının kapsamı daraltılmıştır. Ders programı bir ders kitabı değil, hangi konuların hangi seviyede işlenmesi gerektiği de dahil yazılım testine giriş dersinin temel öğelerini özetlemeye yarayan bir dokümandır. Bu nedenle özellikle:
    - Çoğu durumda metinde örneklere yer verilmemiştir. Eğitim sırasında alıştırmaların yanı sıra örnekleri de sağlamak bir eğitim kurumunun görevidir
    - "Ders programı yazım kontrol listesi" izlenmiştir ve burada her K seviyesinde ÖH'ler için maksimum metin boyutu belirtilir (K1 = maks. 10 satır, K2 = maks. 15 satır, K3 = maks. 25 satır)
- Temel Seviye v3.1.1 ve Çevik v2014 ders programına kıyasla ÖH'lerin sayısında azalma
    - Temel Seviye v3.1.1 (15) ve Çevik 2014 (6) kapsamındaki 21 ÖH'ye kıyasla 14 K1 ÖH'si
    - Temel Seviye v3.1.1 (40) ve Çevik 2014 (13) kapsamındaki 53 ÖH'ye kıyasla 42 K2 ÖH'si
    - Temel Seviye v3.1.1 (7) ve Çevik 2014 (8) kapsamındaki 15 ÖH'ye kıyasla 8 K3 ÖH'si
- Yazılım testi ve ilgili konular hakkında klasik ve/veya saygın kitap ve makalelere daha fazla referans verildi
- Konu 1'de (Testin Temelleri) önemli değişiklikler yapıldı
    - Test becerilerine ilişkin bölüm genişletildi ve geliştirildi
    - Tüm ekip yaklaşımına (K1) ilişkin bölüm eklendi
    - Test bağımsızlığına ilişkin bölüm Konu 5'ten Konu 1'e taşındı
- Konu 2'de (Yazılım Geliştirme Yaşam Döngüsü Boyunca Testler) önemli değişiklikler yapıldı
    - Bölüm 2.1.1 ve 2.1.2 yeniden yazıldı ve geliştirildi, ilgili ÖH'ler değiştirildi

  - Önce-test-et yaklaşımı (K1), shift-left (K2), geçmişe dönük öğeler (K2) gibi uygulamalara daha fazla odaklanıldı
  - DevOps bağlamında testlere ilişkin yeni bölüm eklendi (K2)
  - Entegrasyon testi seviyesi iki ayrı test seviyesine ayrıldı: bileşen entegrasyon testi ve sistem entegrasyon testi
- Konu 3'te (Statik Testler) önemli değişiklikler yapıldı
  - K3 ÖH (bir gözden geçirme tekniği uygulamak) ile birlikte gözden geçirme tekniklerine ilişkin bölüm kaldırıldı
- Konu 4'te (Test Analizi ve Tasarımı) önemli değişiklikler yapıldı
  - Kullanım senaryosu testi kaldırıldı (ancak İleri Düzey Test Analizi ders programında halen mevcuttur)
  - İş birliğine dayalı test yaklaşımına daha fazla odaklanıldı: Test senaryoları türetmek için ATDD kullanımı hakkında yeni K3 ÖH ve kullanıcı hikayeleri ve kabul kriterleri hakkında iki yeni K2 ÖH eklendi
  - Karar testi ve kapsamı, dal testi ve kapsamı ile değiştirildi (birincisi, dal kapsamı uygulamada daha yaygın olarak kullanılmaktadır; ikincisi, farklı standartlar "dalın" aksine kararı farklı şekilde tanımlamaktadır; üçüncüsü, bu, "%100 karar kapsamı %100 komut kapsama yüzdesi anlamına gelir" diyen eski FL2018'in incelikli ancak ciddi bir kusurunu ortadan kaldırmaktadır. Bu cümle karar içermeyen programlar için doğru değildir)
  - Beyaz kutu testine ilişkin bölüm geliştirildi
- Konu 5'te (Test Aktivitelerini Yönetme) önemli değişiklikler yapıldı
  - Test stratejileri/yaklaşımları ile ilgili bölüm kaldırıldı
  - Test eforunu tahmin etmeye yönelik tahminleme teknikleri hakkında yeni K3 ÖH eklendi
  - Test yönetiminde Çevik ile ilgili iyi bilinen kavram ve araçlara daha fazla odaklanıldı: döngü ve sürüm planlaması (K1), test piramidi (K1) ve test çeyrekleri (K2)
  - Risk yönetimine ilişkin bölüm dört ana aktivitenin tanımlanmasıyla daha iyi yapılandırılmıştır: risk belirleme, risk değerlendirmesi, risk azaltma ve risk gözetimi
- Konu 6'da (Test Araçları) önemli değişiklikler yapıldı
  - Bazı test otomasyonu sorunlarına ilişkin içerikler temel seviye için fazla ileri düzeyde olduğundan azaltıldı. Araç seçimi, pilot projelerin gerçekleştirilmesi ve araçların organizasyonda kullanıma alınmasıyla ilgili bölüm kaldırıldı

11. Dizin