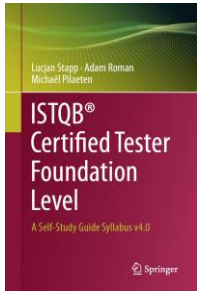
Yazılım Test ve Doğrulama

Dr. Öğr. Üyesi Hakan KEKÜL
Sivas Cumhuriyet Üniversitesi
Teknoloji Fakültesi
Yazılım Mühendisliği

Bu ders notu genel olarak;
ISTQB® Certified Tester Foundation Level - A Self-StudyGuideSyllabusv4.0
Lucjan Stapp • Adam Roman • Michaël Pilaeten - Springer 2023
kitabındaki bilgilerden üretilmiştir.



Bölüm 3. Statik Test

Anahtar Kelimeler

- **Anomaly(Anomali):** Beklentilerden sapan bir durumdur. ISO 24765
- **Dynamic testing (Dinamik test):**Test öğesinin yürütülmesini içeren test. ISO 29119-1'den sonra
- **Formal review (Resmi gözden geçirme):** Resmi olarak belgelenmiş bir çıktıyla tanımlanmış bir süreci takip eden bir inceleme türü. ISO 20246'dan sonra
- **Informal review (Resmi olmayan gözden geçirme):** Tanımlanmış bir süreci takip etmeyen ve resmi olarak belgelenmiş çıktısı olmayan bir inceleme türü
- **Inspection (İnceleme):** Bir çalışma ürünündeki kusurları belirlemek ve inceleme sürecini ve yazılımı iyileştirmek için tanımlanmış ekip rollerini ve ölçümleri kullanan bir tür resmi inceleme gelişme süreci. ISO 20246 Sonra
- **Review (Gözden geçirme):** Bir çalışma ürününün veya sürecinin, kusurları tespit etmek veya iyileştirmeler sağlamak amacıyla bir veya daha fazla kişi tarafından değerlendirildiği bir tür statik test
- **Static analysis(Statik analiz):** Bir bileşeni veya sistemi çalıştırmadan biçimine, yapısına, içeriğine veya belgelerine göre değerlendirme süreci. ISO 24765'ten sonra
- **Static testing(Statik test):** Bir test öğesinin yürütülmesini gerektirmeyen test
- **Technical review (Teknik gözden geçirme):** Bir çalışma ürününün kalitesini inceleyen, spesifikasyonlar ve standartlarla arasındaki tutarsızlıkları belirleyen teknik uzmanlar tarafından yapılan resmi bir inceleme.
- **Walkthrough(Üzerinden geçme):** Bir yazarın, bir çalışma ürünü aracılığıyla inceleme üyelerine yol gösterdiği ve üyelerin olası sorunlar hakkında sorular sorup yorum yaptığı bir inceleme türü. ISO 20246'dan sonra. Eş anlamlı sözcükler: structured walkthrough (yapılandırılmış üzerinden geçme)

3.1 Statik Test Temelleri

FL-3.1.1 (K1) Farklı statik test teknikleriyle incelenebilecek ürün türlerini tanımak

FL-3.1.2 (K2) Statik testin değerini açıklamak

FL-3.1.3 (K2) Statik ve dinamik testleri karşılaştırın

3.1 Statik Test Temelleri

- Statik test, test edilen bileşenin veya sistemin çalıştırılmadığı (yürütülmediği) bir dizi test yöntemi ve tekniğidir.
- Statik testler ayrıca yazılım dışındaki tasarım, dokümantasyon, spesifikasyonlar vb. gibi yürütülemeyen iş ürünlerine de uygulanabilir.
- Statik testin amaçları, özellikle kalite iyileştirmeyi, kusur tespitini ve incelenen çalışma ürününün okunabilirliği, eksiksizliği, doğruluğu, test edilebilirliği veya tutarlılığı gibi özelliklerin değerlendirilmesini içerir.
- Böylece statik test, bir çalışma ürününün hem doğrulanması hem de geçerli kılınması için kullanılabilir.
- Gereksinimlerin eksiksiz, anlaşılır ve test edilebilir olmasını ve kullanıcı hikayeleri söz konusu olduğunda test edilebilir kabul kriterlerini içermesini sağlamak için inceleme teknikleri kullanılabilir.
- Test uzmanları, doğru soruları sorarak önerilen gereksinimleri inceler, sorgular ve iyileştirmeye yardımcı olur.

3.1 Statik Test Temelleri

- Statik testler iki ana teknik grubuna ayrılabilir:
  - Static analysis - Statik analiz
  - Reviews – Gözden geçirmek

- Statik analiz, test edilen iş ürününün araçlar kullanılarak değerlendirilmesini içerir.

- Temel Düzey ders programı statik analiz yöntemlerini ayrıntılı olarak açıklamamaktadır. Daha fazla ayrıntıyı İleri Seviye ders programı “Teknik Test Analisti”nde bulacaksınız.

- Statik analiz tekniklerine örnekler şunları içerir:
  - Kod ölçümü (örneğin, boyutunun veya döngüsel karmaşıklığın ölçülmesi)
  - Kontrol akışı analizi
  - Veri akışı analizi
  - Değişken türlerinin uyumluluğunun kontrol edilmesi, kod yazma standartlarının doğru uygulamasının doğrulanması (örneğin, değişken adlandırma), vb.

3.1 Statik Test Temelleri

- Statik analiz, dinamik testten önce sorunları tanımlayabilir; test senaryoları gerekli olmadığından ve analiz genellikle araçlarla gerçekleştirildiğinden daha az çaba gerektirir.
- Statik analiz genellikle otomatik dağıtım hattının bir adımı olarak sürekli entegrasyon çerçevelerine dahil edilir.
- Her ne kadar büyük ölçüde belirli kod kusurlarını tespit etmek için kullanılsa da, statik analiz aynı zamanda kodun sürdürülebilirliğini, performansını ve güvenlik saldırılarına karşı savunmasızlığını değerlendirmek için de yaygın olarak kullanılır.
- Gözden Geçirmeler (Reviews) çok daha yaygın olarak kullanılan statik test teknikleridir.

3.1.1 Statik Testle İncelenebilecek Çalışma Ürünleri

- Statik test teknikleri hemen hemen her çalışma ürününe uygulanabilir.
- Müfredat, statik teste tabi tutulan bu tür birçok iş ürünü örneğini listelemektedir.
- Aşağıda biraz genişletilmiş bir liste verilmiştir:
  - Her türlü spesifikasyon (iş, işlevsel gereksinimler, işlevsel olmayan gereksinimler, vb.)
  - Epikler, kullanıcı hikayeleri, kabul kriterleri ve çevik projelerde kullanılan diğer belge türleri
  - Mimari tasarım
  - Kaynak kodu
  - Test planları, test prosedürleri, test senaryoları, otomatik test komut dosyaları, test verileri, risk analizi belgeleri ve test başlatma belgeleri dahil her türlü test yazılımı
  - Yerleşik çevrimiçi yardım, sistem operatörü kılavuzları, kurulum talimatları ve sürüm notlarını içeren kullanıcı kılavuzları
  - Web siteleri (içerik, yapı, kullanılabilirlik vb. açısından)
  - Proje belgeleri (ör. sözleşmeler, proje planları, programlar, bütçeler)

3.1.1 Statik Testle İncelenebilecek Çalışma Ürünleri

- Gözden geçirmenin temelde herhangi bir çalışma ürünü için geçerli olabilmesine rağmen, bunun anlamlı olması için gözden geçirmeye katılanların ürünü anlayarak okuyabilmesi ve analiz edebilmesi gerekir.

- Ek olarak, statik analiz durumunda, bu teknikle incelenen iş ürünlerinin, testin yapıldığı resmi bir yapıya sahip olması gerekir.

- Bu tür resmi yapıların örnekleri şunlardır:
  - Kaynak kodu (her program resmi bir dilbilgisine dayalı olarak tanımlanan bir programlama dilinde yazıldığı için)
  - Modeller (örneğin, belirli bir sözdizimine ve bunları oluşturmak için resmi kurallara sahip UML diyagramları)
  - Metin belgeleri (çünkü metin kontrol edilebilir, örneğin dilin gramer kurallarına aykırılığı)

3.1.1 Statik Testle İncelenebilecek Çalışma Ürünleri

- Gözden geçirmenin temelde herhangi bir çalışma ürünü için geçerli olabilmesine rağmen, bunun anlamlı olması için gözden geçirmeye katılanların ürünü anlayarak okuyabilmesi ve analiz edebilmesi gerekir.

- Ek olarak, statik analiz durumunda, bu teknikle incelenen iş ürünlerinin, testin yapıldığı resmi bir yapıya sahip olması gerekir.

- Bu tür resmi yapıların örnekleri şunlardır:
  - Kaynak kodu (her program resmi bir dilbilgisine dayalı olarak tanımlanan bir programlama dilinde yazıldığı için)
  - Modeller (örneğin, belirli bir sözdizimine ve bunları oluşturmak için resmi kurallara sahip UML diyagramları)
  - Metin belgeleri (çünkü metin kontrol edilebilir, örneğin dilin gramer kurallarına aykırılığı)

- Statik analiz çoğunlukla resmi sistem mimarisi modelleri veya gereksinimleri (örneğin, Z veya UML gibi spesifikasyon dillerinde yazılmış) gibi resmileştirilmiş iş ürünlerine uygulanır.

- Doğal dilde yazılmış belgeler için statik analiz, örneğin okunabilirlik, sözdizimi, dil bilgisi, noktalama işaretleri veya yazım denetimi yapılmasını içerebilir.

3.1.2 Statik Testin Değeri

- Statik test genellikle ucuz değildir (ürünün manuel olarak kontrol edilmesini gerektirir), ancak doğru şekilde yapılırsa etkili ve verimlidir.
- Bunun nedeni, daha sonraki aşamalarda tespit edilmesi zor olan kusurların ve sorunların SDLC'de çok erken bulunmasına olanak sağlamasıdır.
- Bu genellikle tasarım kusurlarını ifade eder.
- Bu tür kusurlar genellikle çok sinsi olur, çünkü ilk aşamalarda tespit edilemeyen bir tasarım hatası sonraki aşamalara çok kolay yayılır ve hatalı bir tasarım üzerinden hatalı bir uygulama yaratılır.
- Sorun genellikle sistem veya kabul testi aşamasında ortaya çıkar ve bu tür bir tasarım hatasını düzeltmenin çok pahalı olabileceği durumlara yol açar.

3.1.2 Statik Testin Değeri

- Dinamik testin yürütülmesinden önce iyi yürütülen statik test, diğer kusurların kaynağı olabilecek sorunların hızlı (ve oldukça ucuz) ortadan kaldırılmasıyla sonuçlanır.
- Sonuç olarak, dinamik testte daha az sorun tespit edilir, dolayısıyla dinamik testin toplam maliyeti, statik testin yapılmamasına göre daha düşüktür.
- Statik testin bu şekilde kullanılması erken test ve shift-left ilkesiyle uyumludur.
- Statik test, kaliteyi değerlendirme ve incelenen iş ürününe güven oluşturma fırsatı sağlar.
- Paydaşlar, belgelenen gereksinimlerin gerçek ihtiyaçlarını tanımlayıp tanımlamadığını değerlendirebilirler.
- Statik test, SDLC'nin başlarında gerçekleştirilebildiği için, statik teste dahil olan paydaşlar arasında ürün ve gereksinimlerine ilişkin ortak bir anlayış oluşturulur.
- Bu ortak anlayış aynı zamanda iletişimi de geliştirir. Bu nedenle incelemelerin, incelenen iş ürününe ilişkin en geniş, en çeşitli bakış açılarını temsil eden çeşitli paydaşları dahil etmesi önerilir.

3.1.2 Statik Testin Değeri

- Koddaki hatalar statik test kullanılarak dinamik test kullanıldığında olduğundan daha verimli bir şekilde tespit edilebilir ve düzeltilebilir, çünkü dinamik testte bir hatanın ortaya çıkması genellikle hatanın nedeninin sıkıcı bir şekilde analiz edilmesini ve hataya neden olan kusurun belirlenmesi için çok fazla zaman harcanmasını gerektirir.

- Statik testin bu verimliliği genellikle hem kodda kalan hataların azalmasına hem de genel iş yükünün azalmasına neden olur.

- Statik tekniklerin etkinliği ampirik olarak kanıtlanmıştır.

- Denetim durumunda, kusur giderme verimliliği ortalama %85 oranında artar!

3.1.2 Statik Testin Değeri

- Aşağıda statik testin faydalarının bir listesi bulunmaktadır:
  - Dinamik test başlamadan önce bile erken, etkili hata tespiti (ve potansiyel kaldırma) - çalışan bir yazılım prototipi oluşturulmadan önce bile test etme yeteneği.
  - Sonraki dinamik testlerde tespit edilmesi zor olan kusurların belirlenmesi. Bu özellikle tasarım ve mimari kusurlar için geçerlidir.
  - Mümkün olmayan (ulaşılamaz) kod, kullanılmayan kod, koddaki tasarım kalıplarının hatalı kullanımı veya kullanılmaması veya çalıştırılamayan iş ürünlerindeki her türlü kusurun tespiti gibi dinamik testlerde tespit edilmesi mümkün olmayan kusurların belirlenmesi, dokümantasyon gibi.
  - Gereksinim spesifikasyonu veya mimari tasarımı gibi belgelerdeki belirsizlikleri, çelişkileri, eksiklikleri, gözden kaçanları, gereksiz öğeleri veya tutarsızlıkları tespit ederek tasarım ve kodda kusurların oluşmasını önlemek.
  - Programlama çalışmalarının verimliliğinin arttırılması. Tasarımın ve kodun sürdürülebilirliğinin iyileştirilmesi, örneğin bunların yazılması için tek tip bir standart getirilerek sağlanabilir. Bu, kodun yalnızca yazar tarafından değil aynı zamanda diğer geliştiriciler tarafından da değiştirilmesini kolaylaştırır ve kod değiştirilirken bir kusur ortaya çıkma şansı azalır.
  - Testler (özellikle dinamik testler) dahil olmak üzere yazılım geliştirmenin maliyetini ve süresini azaltmak.
  - Bakım aşamasındaki maliyetleri azaltarak yazılım geliştirme döngüsü boyunca kalite maliyetini düşürmek ve ayrıca yazılımdan yararlanma aşamasındaki arızaları azaltmak.
  - İnceleme toplantıları da dahil olmak üzere incelemelere katılım yoluyla ekip üyeleri arasındaki iletişimi geliştirmek.

3.1.2 Statik Testin Değeri

- Test için katlanılan maliyetlerin değerlendirilmesinde “kalite maliyeti” kavramı devreye girmektedir.
- Bu, aşağıdakilerin maliyetlerinden oluşan, kaliteli faaliyetler için katlanılan toplam maliyettir:
  - Önleyici faaliyetler (ör. eğitim maliyetleri)
  - Tespit (ör. test maliyeti)
  - Dahili arızalar (ör. üretimde bulunan kusurları düzeltme maliyeti)
  - Dış arızalar (ör. kullanıcılar tarafından bulunan saha kusurlarını düzeltme maliyeti)
- Uygulaması maliyetli olsa da, genel kalite maliyetleri genellikle incelemelerin yapılmadığı duruma göre çok daha düşüktür, çünkü projenin ilerleyen aşamalarında kusurların giderilmesi için daha az zaman ve çaba harcanması gerekir.
- İnceleme sürecindeki katılımcılar aynı zamanda incelenen ürüne ilişkin daha iyi bir ortak anlayıştan da yararlanır.

3.1.2 Statik Testin Değeri
İncelemelerin örnek ekonomik analizi

- Bir kuruluşta statik test uygulamasının işe yarayıp yaramayacağını gösteren basitleştirilmiş bir ekonomik simülasyonu düşünün.
- Senaryo A'da ekip yalnızca dinamik testler gerçekleştirecektir.
- Senaryo B'de dinamik testten önce statik test uygulanacaktır.
- Aşağıdaki model çok basittir ve yalnızca statik testin yazılım geliştirme ve bakımının toplam maliyetini nasıl azaltabileceğini göstermeyi amaçlamaktadır.

3.1.2 Statik Testin Değeri

İncelemelerin örnek ekonomik analizi

- Model varsayımları:
  - Test ekibi altı kişiden oluşur.
  - Bir test uzmanı maaşının aylık maliyeti 7000$'dır.
  - Statik testler ve dinamik testlerin her biri 4 ay sürer ve test ekibinin tamamı katılır. Dinamik testlerden önce (varsa) statik testler yapılır.
  - Toplam yazılım hatası sayısı 130'dur.
  - Statik test mevcut tüm kusurların %50'sini bulur.
  - Dinamik test mevcut tüm kusurların %80'ini bulur. Geri kalan %20'lik kısım ise yazılım piyasaya sürüldükten sonra müşteri tarafından keşfedilen saha kusurlarıdır (bu değerler geçmiş verilerden elde edilmiştir).
  - Statik testlerde bulunan bir kusurun düzeltilmesi ortalama 500$'a mal olur (endüstri verileri).
  - Dinamik testlerde bulunan bir kusurun düzeltilmesi ortalama 1800$'a mal olur (geçmiş veriler).
  - Bir saha kusurunun (yazılım piyasaya sürüldükten sonra kullanıcı tarafından bulunan) onarılmasının maliyeti 12.600 ABD dolarıdır (geçmiş veriler).

3.1.2 Statik Testin Değeri
İncelemelerin örnek ekonomik analizi

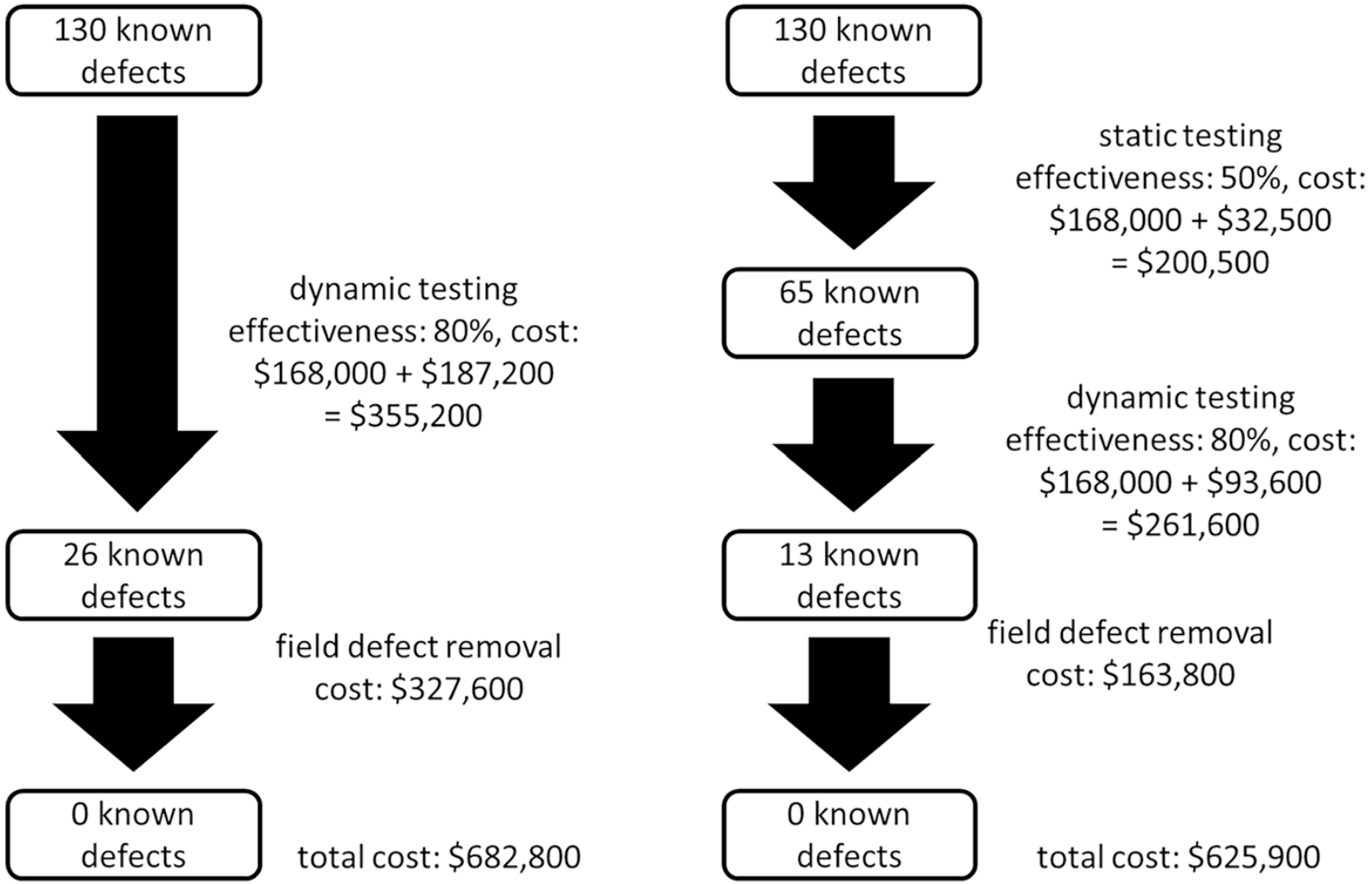
- Senaryo A: Statik test olmadan
  - Dinamik testte tespit edilen kusur sayısı: %80 * 130 = 104
  - Saha kusurlarının sayısı: %20 * 130 = 26
  - Dinamik testlerin maliyeti: 6 * 7000 $ * 4 = 168.000 $
  - Yayınlanmadan önce kusur giderme maliyeti: 104 * 1800$ = 187.200 $
  - Sürümden sonra kusurları gidermenin maliyeti: 26 * 12.600 $ = 327.600 $
  - Toplam kalite maliyeti: 168.000 $ + 187.200 $ + 327.600 $ = 682.800 $

3.1.2 Statik Testin Değeri
İncelemelerin örnek ekonomik analizi

- Senaryo B: Statik test ile
  - Statik testte tespit edilen kusur sayısı: %50 * 130 = 65
  - Dinamik testte tespit edilen kusur sayısı: %80 * (130 - 65) = 52
  - Saha kusurlarının sayısı: 130 - (65 + 52) = 13

  - Statik testin maliyeti: 6 * 7000 $ * 4 = 168.000 $
  - Dinamik testin maliyeti: 6 * 7000 $ * 4 = 168.000 $
  - Statik testte bulunan kusurları gidermenin maliyeti: 65 * 500 $ = 32.500 $
  - Sürümden önce kusur gidermenin maliyeti: 52 * 1800 $ = 93.600 $
  - Sürümden sonra kusurları gidermenin maliyeti: 13 * 12.600 $ = 163.800 $
  - Toplam kalite maliyeti: 168.000 $ + 168.000 $ + 32.500 $ + 93.600 $ + 163.800 $= 625.900 $

3.1.2 Statik Testin Değeri
İncelemelerin örnek ekonomik analizi

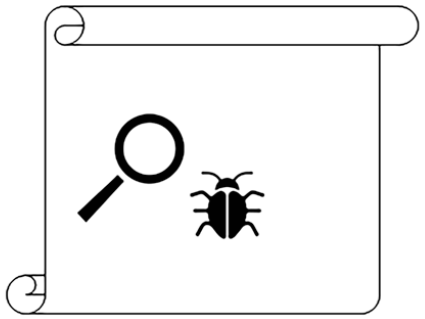


3.1.2 Statik Testin Değeri

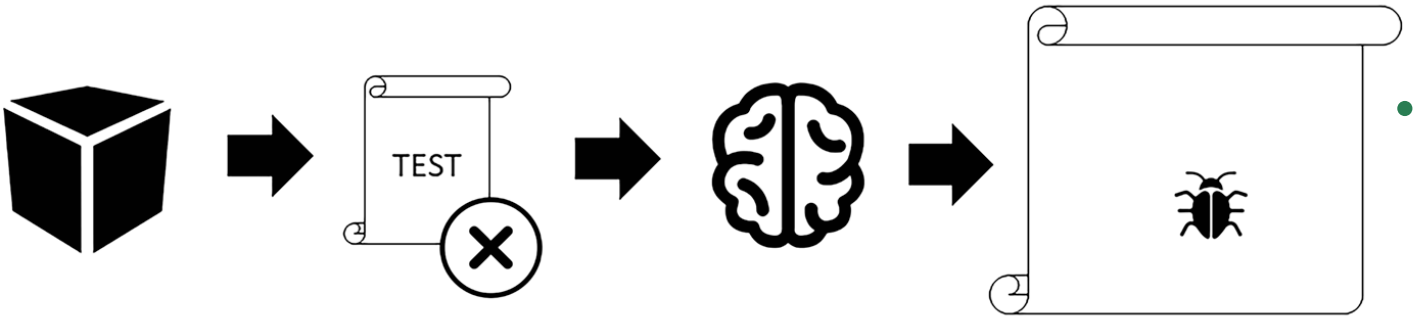
İncelemelerin örnek ekonomik analizi

- Yüksek incelme maliyetleri nedeniyle, Senaryo B'de yayın öncesi kalite maliyetleri daha yüksektir (462.100 $ karşı 355.200 $).
- Ancak, incelemelerin kullanılması sayesinde Senaryo B'deki saha kusurlarının yarı yarıya azalması nedeniyle kazanç, sürümden sonra elde edilir.
- Dolayısıyla toplam maliyetin Senaryo B'de daha düşük olduğu ortaya çıkıyor.
- Senaryo A ve B arasındaki kesin maliyet farkı 682.800 $ - 625.900 $ = 56.900 $'dır.
- Elbette analizde tahmin hatalarının yanı sıra test ekibinin statik testi gerçekleştirmek için ek dört aya daha ihtiyaç duyduğu gerçeğini de hesaba katmalıyız.
- Bununla birlikte analiz, statik test kullanımının yalnızca ürünün nihai kalitesini önemli ölçüde artırmakla kalmayıp (13'e karşılık 26 saha hatası) aynı zamanda geliştirme maliyetlerini de önemli ölçüde azalttığını göstermektedir.
- Yukarıdaki simülasyonda yaklaşık 57.000 $ tasarruf ettik, dolayısıyla Senaryo B'deki maliyetler Senaryo A'dakilerden yaklaşık %8,3 daha az.

3.1.3 Statik Test ile Dinamik Test Arasındaki Farklar



Static testing: defect detection



Dynamic testing: causing failure, analysis, defect localization

- Hem statik test hem de dinamik test aynı hedefe sahiptir; ürün kalitesini değerlendirmek ve kusurları mümkün olan en kısa sürede tespit etmek.
- Bu faaliyetler, farklı türdeki kusurların tespitine olanak sağladığından tamamlayıcı niteliktedir.
- Şekil 3.2, statik test (şeklin üst kısmı) ile dinamik test (şeklin alt kısmı) arasındaki temel farkı sembolik olarak göstermektedir.
- Statik testlerle doğrudan iş ürününde bir kusur bulabiliyoruz. Arıza bulmuyoruz, çünkü statik tekniklerde tanım gereği yazılımı çalıştırmıyoruz ve arıza yalnızca çalışan sistemin bir sonucu olarak ortaya çıkabilir.

3.1.3 Statik Test ile Dinamik Test Arasındaki Farklar

- Dinamik test durumunda, çoğu zaman arızalı yazılımın ilk işareti başarısızlıktır (bir testin başarısız olmasının sonucu).
- Bir arıza gözlemlendiğinde, bu arızaya neden olan kusuru bulmak için kaynak kodun analiz edildiği bir hata ayıklama süreci başlatılır.
- Bazen bir iş ürünündeki kusur, arızaya yol açmadığı için çok uzun süre gizli kalabilir.
- Ayrıca, üzerinde bulunduğu yol nadiren test ediliyor olabilir veya erişilmesi zor olabilir, bu da onu tespit edecek dinamik testin tasarlanması ve yürütülmesini kolaylaştırmaz.
- Statik test, çok daha az çaba harcayarak bir kusur bulabilir.
- Öte yandan, bellek sızıntıları gibi dinamik testlerle tespit edilmesi statik testlerden çok daha kolay olan kusurlar da vardır.

3.1.3 Statik Test ile Dinamik Test Arasındaki Farklar

- Diğer bir fark, statik testin iş ürünlerinin tutarlılığını ve iç kalitesini artırmak için kullanılabilmesi, dinamik testin ise esas olarak dışarıdan görünen davranışlara odaklanmasıdır.
- Ayrıca kaynak kodu veya dokümantasyon gibi yürütülemeyen çalışma ürünlerine yalnızca statik testler uygulanabilir.
- Dinamik test ise yalnızca çalışan bir iş ürününe, yani çalışan bir yazılıma karşı gerçekleştirilebilir.
- Bu gerçeğin bir sonucu olarak dinamik test, çalışan programa bağlı olduğundan statik testte ölçülmesi imkansız olan bazı kalite özelliklerini ölçmek için kullanılabilir.
- Bunun bir örneği, belirli bir kullanıcı isteği için sistemin yanıt süresini ölçen performans testleri olabilir.

3.1.3 Statik Test ile Dinamik Test Arasındaki Farklar

- Statik testlerle tespit edilmesi ve düzeltilmesi dinamik testlere göre daha kolay ve daha ucuz olan tipik kusurların örnekleri şunlardır:
  - Gereksinimlerdeki kusurlar (tutarsızlıklar, belirsizlikler, atlamalar, çelişkiler, yanlışlıklar, ihmaller, tekrarlar, gereksiz öğeler gibi)
  - Tasarım kusurları (örneğin, verimsiz algoritmalar veya veritabanı yapıları, yüksek bağlantı, düşük uyum,zayıf kod modülerizasyonu)
  - Spesifik kod kusurlarının türleri (örneğin, tanımlanmamış değerleri olan değişkenler, bildirilen ancak asla kullanılmayan değişkenler, erişilemeyen kod, yinelenen kod, çok yüksek zaman veya bellek karmaşıklığına sahip, verimsiz uygulanan algoritmalar)
  - Standartlardan sapmalar (örneğin, kod geliştirme standartlarıyla uyumluluk eksikliği)
  - Yanlış arayüz özellikleri (örneğin, çağıran ve çağrılan sistemlerde farklı ölçü birimlerinin kullanılması, bir API işlev çağrısına iletilen parametrelerin türü veya sırası)
  - Güvenlik açıkları
  - İzlenebilirlik veya kapsamdaki boşluklar veya yanlışlıklar (örneğin, belirli bir kullanıcı hikayesi için kabul kriterleriyle eşleşen testlerin olmaması)

3.1.3 Statik Test ile Dinamik Test Arasındaki Farklar

- Örnek
  - Test ekibi, yazılımın müşteriye sunulmasından sonra bakımının kolaylığını ve maliyetini değerlendirmek istiyor.
  - Bunu yapmak için statik analiz tekniklerini kullanmayı amaçlamaktadır.
  - Kaynak kodunu ölçmek için kullanılacak üç ölçüm belirlenmiştir:

Component	LOC	COMMENT	CC
main	129	0%	7
split	206	1%	12
to_lower	62	1%	6
to_upper	63	3%	6
merge	70	0%	15
config	42	15%	8
stdio	243	2%	21

- LOC—bir bileşendeki yürütülebilir kod satırlarının sayısı
- COMMENT—bir bileşendeki açıklamalı kod satırlarının yüzdesi
- CC—bir bileşenin döngüsel karmaşıklığı

3.1.3 Statik Test ile Dinamik Test Arasındaki Farklar

- Statik analiz aşağıdaki sonuçlara yol açar:
  - Bileşenler oldukça küçüktür. Bunların en büyüğü olan stdio, 243 satır koda sahiptir. Bu nedenle bileşenlerin boyutu, bakım kolaylığı açısından büyük bir sorun olmamalıdır.
  - Kod yeterince yorumlanmamıştır. Kodun %15'inin yorumlandığı config modülü hariç diğer tüm modüllerde yorum yapılan satırların yüzdesi %0 ile %3 arasındadır. main ve merge modülleri hiçbir şekilde yorum içermez. Bu gerçek kesinlikle geliştiricilerin dikkatine sunulmalıdır.
  - Üç bileşen, split, merge ve stdio, yüksek (10'un üzerinde) döngüsel karmaşıklığa sahiptir. Bunların analiz edilmesi gerekiyor. Bu, özellikle karar bildirimlerinin yoğunluğunun çok yüksek olduğu birleştirme için geçerlidir: yaklaşık beş kod satırı için tek bir karar, bu da kaynak kodun çok zayıf okunabilirliğine işaret edebilir. Bu bileşenlerin yeniden düzenlenmemesi, gelecekte bunların bakımını zorlaştırabilir.

3.2 Geri Bildirim ve Gözden Geçirme Süreci

FL-3.2.1 (K1) Paydaşların erken ve sık geri bildiriminin faydalarını tanımlayın
FL-3.2.2 (K2) Gözden geçirme sürecinin faaliyetlerini özetleyin
FL-3.2.3 (K1) Gözdem geçirme gerçekleştirirken ana rollere hangi sorumlulukların atandığını hatırlayın
FL-3.2.4 (K2) Farklı gözden geçirme türlerini karşılaştırın
FL-3.2.5 (K1) Başarılı bir gözden geçirmeye katkıda bulunan faktörleri hatırlayın

3.2 Geri Bildirim ve İnceleme Süreci

- Bu bölümde ayrıntılı olarak tartışacağımız konular:
  - Erken ve sık müşteri geri bildiriminin avantajları
  - İş ürünü gözden geçirme süreci
  - Resmi gözden geçirmede roller ve sorumluluklar
  - Gözden geçirme türleri
  - Gözden geçirmedeki başarı faktörleri
- İş ürünlerini gözden geçirme tekniklerini, Temel Düzey müfredat v3.1'de mevcuttu ancak yeni müfredatta (v4.0) kaldırıldı. Ancak, bu konu fazladan bir bölüm olarak anlatılmaktadır. Çünkü bir çalışma ürününü gözden geçirme etkinliğine pratikte nasıl yaklaşılacağına dair önemli konu.

3.2.1 Erken ve Sık Paydaş Geri Bildiriminin Yararları

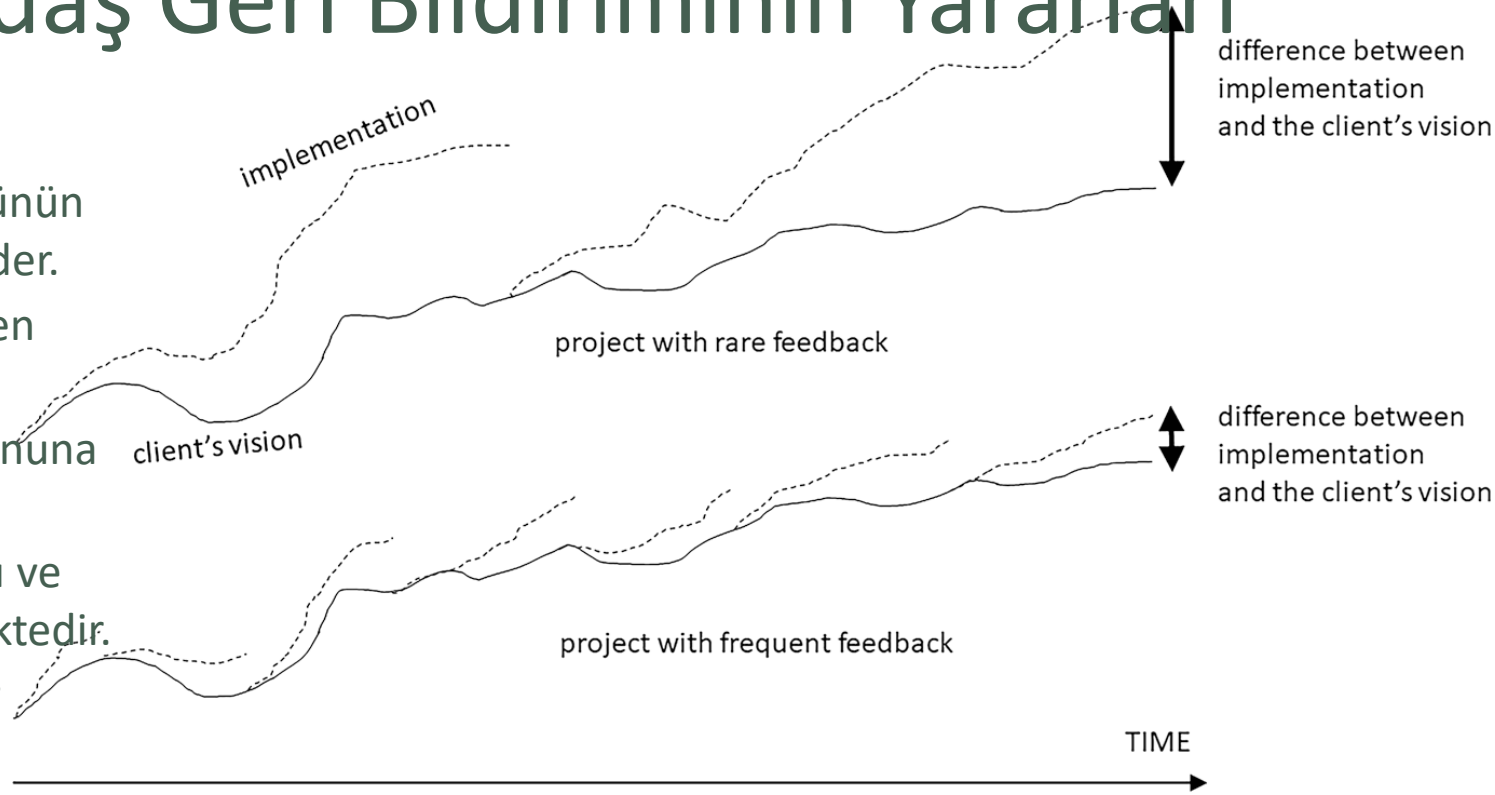
- Gözden geçirmeler, ekibe erken geri bildirim sağlamanın bir biçimidir çünkü geliştirme döngüsünün erken aşamalarında yapılabilirler.
- Ancak geri bildirim yalnızca gözden geçirmelerle sınırlı kalmamalı, projenin geliştirme döngüsü boyunca yaygın bir uygulama olmalıdır.
- Geri bildirimler test uzmanlarından veya geliştiricilerden gelebilir ancak müşterinin kendisinden gelen geri bildirimler de ekip için aynı derecede önemli olacaktır.
- Erken ve sık geri bildirim, potansiyel kalite sorunlarının hızlı bir şekilde bildirilmesine olanak tanır ve ekibin soruna hızlı bir şekilde yanıt vermesini sağlar.
- Yazılım geliştirme sırasında paydaşların katılımı azsa, geliştirilmekte olan ürün orijinal veya mevcut vizyonunu karşılamayabilir.
- Paydaşların beklentilerinin karşılanmaması, maliyetli yeniden çalışmalara, son teslim tarihlerinin kaçırılmasına, suçlama oyunlarına ve hatta projenin tamamen başarısız olmasına neden olabilir.

3.2.1 Erken ve Sık Paydaş Geri Bildiriminin Yararları

- Yazılım geliştirme sırasında paydaşlardan sık sık geri bildirim alınması, gereksinimlerle ilgili yanlış anlaşılmaları önleyebilir ve gereksinim değişikliklerinin daha erken anlaşılmasını ve uygulanmasını sağlayabilir.
- Bu, ekibin ne inşa ettiklerini daha iyi anlamasına yardımcı olur.
- Paydaşlara en fazla değeri sağlayan ve üzerinde anlaşılan riskler üzerinde en olumlu etkiye sahip olan işlevlere odaklanmalarına olanak tanır.

3.2.1 Erken ve Sık Paydaş Geri Bildiriminin Yararları

- Şekil sembolik olarak sık geri bildirimin faydasını göstermektedir.
- Düz çizgi müşterinin vizyonunu, kesikli çizgi ise ürünün uygulamada gerçekte nasıl göründüğünü temsil eder.
- Bu iki vizyon zamanla giderek daha fazla birbirinden ayrılıyor.
- Geri bildirim anlarında uygulama müşterinin vizyonuna göre ayarlanır.
- Üstteki grafik, müşteriyle temasın düzensiz olduğu ve yalnızca iki kez gerçekleştiği bir durumu göstermektedir.
- Sonuçta müşterinin vizyonundan oldukça farklı bir uygulama ortaya çıktığı görülüyor.



- Alttaki grafik, müşterinin ekibe sıklıkla geri bildirim sağladığı bir durumu temsil ediyor. Bu, müşterinin vizyonu ile fiili uygulama arasında daha az fark oluşmasına neden olur, çünkü müşteriyle yapılan toplantılar arasında üstteki tablodaki duruma göre çok daha az zaman geçer.
- Sonuç olarak nihai ürün müşterinin vizyonuyla çok daha uyumlu hale gelir.
- Ayrıca, ürünün çok daha büyük ölçüde değiştirilmesi ve ayarlanması gerekeceğinden, seyrek geri bildirim durumunda ürünü müşterinin görüşüne göre "ayarlamanın" maliyetinin çok daha yüksek olacağı unutulmamalıdır.

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

- Gözden geçirme türü, resmileştirme düzeyi veya hedefleri farklılık gösterir, ancak tüm gözden geçirme türleri benzer aşamalara veya faaliyet gruplarına sahip olma eğilimindedir.
- ISO 20246 standardı Yazılım ve sistem mühendisliği - İş ürünü incelemeleri, belirli bir gözden geçirme sürecini belirli bir duruma göre uyarlamak için yapılandırılmış ancak esnek bir çerçeve sağlayan genel bir inceleme sürecini tanımlar.
- Gerekli gözden geçirme daha resmi ise, sürecin daha fazla görevi veya öğesi olacaktır. ISO/IEC 20246'da açıklanan genel bir inceleme süreci Şekilde gösterilmektedir.
- Birçok iş ürününün boyutu, bunların tek bir incelemenin kapsamına giremeyecek kadar büyük olabileceği anlamına gelir. Bu gibi durumlarda inceleme süreci genellikle çalışma ürününü oluşturan ayrı parçalara birden çok kez uygulanır.
- Temel Düzey ders programı, Şekilde gösterilen ve iş ürününün gözden geçirilmesi sürecinde meydana gelebilecek beş genel aktivite türünü tanımlar.

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

Planlama-Planning

- Planlama, gözden geçirmenin kapsayacağı işin kapsamını tanımlar.
- Planlama, kim, ne, nerede, ne zaman ve neden gibi soruları yanıtlayarak gözden geçirmenin sınırlarını belirler.
- Gözden geçirmenin amacı (neden) ve hangi belgelerin incelemeye tabi tutulacağı tanımlanır.
- Gözden geçirilecek kalite özellikleri tanımlanır (ne).
- Gözden geçirmeyi yürütmek için gereken iş miktarı tahmin edilir ve sürecin belirli aşamalarının zaman çerçevesi ve yeri tanımlanır (nerede ve ne zaman).
- Gözden geçirmenin amacına bağlı olarak, inceleme türü, kullanılacak roller, faaliyetler ve kontrol listeleri (gerekirse) ile birlikte tanımlanır.
- Gözden geçirmeye katılacak kişiler seçilir ve roller atanır (kim).
- İnceleme/denetim gibi resmi gözden geçirmeler için resmi giriş ve çıkış kriterleri tanımlanır ve bu aşamanın sonunda giriş kriterlerinin karşılandığı ve gözden geçirmenin bir sonraki aşamaya geçebileceği de doğrulanır.

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

Gözden Geçirmenin Başlatılması - Review initiation

- Gözden geçirmenin başlatılmasının bir parçası olarak, gözden geçirilecek çalışma ürünü, kontrol listeleri, prosedür bildirimleri ve kusur raporu gibi şablonlar diğer tüm gerekli materyallerle birlikte gözden geçirme katılımcılarına (planlama aşamasında seçilen) gönderilir.
- Gözden geçirmeyi yapanların görevlerini tamamlamak için yeterli zamana sahip olmaları için tüm materyaller mümkün olduğu kadar erken dağıtılmalıdır.
- Gerektiğinde katılımcılara gözden geçirmenin nasıl gerçekleşeceği ve rollerinin ne olduğu konusunda açıklamalar yapılır.
- Katılımcıların herhangi bir sorusu varsa, başlatma aşamasında yanıtlar ve açıklamalar sağlanır, böylece herkes ne yapması gerektiğinin bilincinde olur ve bu faaliyetlerin zaman dilimini bilir.
- Bu aşamada gözden geçirme eğitimi düzenlemek mümkündür.
- Katılımcılar gözden geçirmeler konusunda deneyimsiz olduğunda ve sürecin sorunsuz ve verimli bir şekilde ilerlemesini istediğinizde bu mantıklıdır.
- Resmi gözde geçirmeşer için, bireysel katılımcılara kapsam ve bireysel rollerini ve sorumluluklarını açıklamak için bir başlangıç toplantısı da yapılabilir.
- Başlatma aşamasının amacı, katılan herkesin hazırlıklı olduğundan ve başlamaya hazır olduğundan emin olmaktır.

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

Bireysel Gözden Geçirme – Individual Review

- Bireysel gözden geçirme yani bireysel hazırlık.
- Bu, herhangi bir gözden geçirmenin temel ve merkezi aşamasıdır.
- Bu aşamada, önemli faaliyetler gerçekleştirilir, yani iş ürününün gözden geçirenler tarafından fiilen gözden geçirilmesi, belirli teknikleri kullanılarak gerçekleştirilir.
- Bu faaliyetlerin bir parçası olarak iş ürününün tamamı veya bir kısmı gözden geçirilir.
- Gözden geçirenler, çalışma ürününün incelenmesi sırasında yapılan yorumları, soruları, tavsiyeleri, endişeleri ve ilgili gözlemleri kaydeder.
- Tüm gözden geçirme aşamaları arasında, bireysel gözden geçirme aşaması sorunların en büyük yüzdesini tespit eder.
- Tanımlanan sorunlar genellikle bir hata yönetimi veya gözden geçirme destek aracı tarafından desteklenen bir sorun günlüğünde belgelenir.
- ISO/IEC 20246 standardına göre, bu süreç adımı isteğe bağlıdır ve izlenecek yollar veya resmi olmayan gözden geçirmeler gibi belirli gözden geçirme türleri için gerçekleştirilemeyebilir.
- Ancak birçok kişi bunun tüm inceleme sürecindeki en önemli aktivite olduğu görüşünü benimser.

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

İletişim ve analiz – Communication and analysis

- Bir gözden geçirme toplantısı planlanmamışsa, bulunan sorunlar, bu sorunların analiziyle birlikte doğrudan kişilere (örneğin, incelenen çalışma ürününün yazarına) iletilir.
- Gözden geçirme toplantısı gözden geçirme planına dahil edilmişse, önceden gözden geçirilmeleri için toplantıdan önce toplu olarak tüm katılımcılara gönderilir, böylece gözden geçirme toplantısı daha verimli çalışacaktır.
- Toplantı, bildirilen sorunların (anormalliklerin) analizini içerir.
- Her anormalliğin mutlaka bir kusur olduğu ortaya çıkmayabileceğinden, gerçek bir sorun (kusur) mu yoksa yalnızca görünen bir sorun mu (yanlış pozitif) olduğuna karar vermek için tüm bulgular toplantıda dikkatle gözden geçirilmelidir.
- Anormalliğin kusur olduğu ortaya çıkarsa, bunları düzeltmekten sorumlu olanlar atanır ve durum, öncelik, ciddiyet gibi parametreler tanımlanır.
- Bu eylemler bir gözden geçirme toplantısında veya (böyle bir toplantı yoksa) bireysel olarak gerçekleştirilir.
- Bildirilen anormallikler için en sık kullanılan durumlar şunlardır:
  - Sorun reddedildi
  - Sorun herhangi bir işlem yapılmadan kaydedildi
  - Sorun iş ürünü yazarı tarafından çözülecek
  - Sorun daha ileri analiz sonucunda güncellendi
  - Sorun dış bir paydaşa atandı

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

İletişim ve analiz – Communication and analysis

- Bu aşama aynı zamanda planlama aşamasında tanımlanan kalite özelliklerinin düzeyini de gözden geçirmek olarak değerlendirilir ve belgelendirilir.
- Son olarak, incelemenin sonuçları, gözden geçirilen çalışma ürünüyle ne yapılacağına karar vermek için çıkış kriterlerine göre değerlendirilir.
- Karar örnekleri şunları içerir:
  - İş ürününün kabulü (genellikle hiçbir sorun olmadığında veya yalnızca az sayıda önemsiz sorun tespit edildiğinde).
  - Tanımlanan sorunlara göre iş ürününü güncelleyin.
  - İş ürünü tekrar gözden geçirilmelidir (genellikle çok sayıda sorun olması ve değişikliklerin kapsamının geniş olması nedeniyle).
  - İş ürününün reddedilmesi (en nadir karar türüdür, ancak bu da gerçekleşebilir).

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

Düzeltme ve Raporlama – Fixing and reporting

- Bu gözden geçirmenin son aşamasıdır.
- Tespit edilen hatalara ilişkin değişiklik gerektiren hata raporları oluşturur.
- İncelenen çalışma ürününün yazarının bu aşamada kusur giderme işlemini gerçekleştirmesi beklenmektedir.
- Bu, incelenen iş ürününde tespit edilen kusurların ilgili kişilere veya ilgili ekibe bildirilmesiyle gerçekleştirilir.
- Son olarak değişiklikler onaylandığında bir gözden geçirme raporu oluşturulur.
- Daha resmi gözden geçirme türleri için ayrıca, gözden geçirmenin etkililiğini test etmek amacıyla çeşitli türde önlemler kullanılır.
- Ayrıca çıkış kriterlerine göre de kontrol edilir ve çıkış kriterlerinin karşılandığı belirlendikten sonra iş ürünü kabul edilir.
- Bir iş ürünü gözden geçirme sonuçları, gözden geçirmenin türüne ve resmileştirme derecesine bağlı olarak değişebilir.

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

Örnek

- Bir ekip, ürettikleri bir çevrimiçi RPG bilgisayar oyununda beceri ağacı olarak adlandırılan şeyin gözden geçirmesini yapmaya karar verdi.
- Test lideri, gözden geçirme katılımcısı olarak beş kişiyi seçer: bir geliştirici, bir test uzmanı ve ekibin dışından, dış paydaşlar (oyuncular) olarak görev yapan üç kişi.
- Gerçekleştirilen gözden geçirme türünün bireysel, gayri resmi bir gözden geçirme olmasına karar verilmiştir.
- Ekip lideri katılımcılara beceri ağacını açıklayan bir belgeyi, kontrol edilecek özelliklerin bir kontrol listesini ve bir sorun günlüğü şablonunu dağıtır.
- Her katılımcı, önceden belirlenen bir zamanda belgeyi inceler, sorun günlüğü formunu doldurur ve ardından ekip liderine gönderir.
- Ekip lideri formları birleştirir, gereksiz hataları giderir ve bu şekilde oluşturulan sorunların listesini geliştiricilere gönderir, aynı zamanda tespit edilen hataları hata raporlarına dönüştürür ve bunları hata yönetimi aracına kaydeder.

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

No.	Page/line	Defect category	Severity	Defect type	Source of defect	Description, comments
...	.....	.....	.....	.....	.....	.....

Defect categories: Missing, Defect, Redundancy

Sorun günlüğü şablonu

Gözden geçirenlerden biri tarafından doldurulmuş bir sorun günlüğü

No.	Page/line	Defect category	Severity	Defect type	Description, comments
1	Fig. 1	Defect	Small	Syntax	"Electric lighnting" instead of "Electric lightning"
2	Fig. 1	Defect	Large	Logic	The element following "Electric Lightning" should be the "Electric Storm" skill, not "Firestorm"
3	Fig. 1	Missing	Large	Logic	The skill "Blinding the opponent" is missing, the introduction of which to the skill tree was agreed at the Feb 4th meeting

3.2.2 Süreç Faaliyetlerini Gözden Geçirme

Örnek

- Gözden geçirenlerden biri tarafından bildirilen kusurların bir örneği Tabloda gösterilmektedir.
- Gözden geçirmeyi yapan kişi kontrol listesini kullanmadığı için "Kusur kaynağı" sütununa ihtiyaç duyulmadığından bu sütun sorun günlüğünden kaldırıldı.
- Yazarlar bulunan kusurları düzeltir ve bunu kusur yönetimi aracına kaydeder.
- Gözden geçirme lideri tüm kusurların giderildiğini doğrular.
- Gözden geçirme lideri daha sonra aşağıdakilerle ilgili bilgileri içeren bir özet gözden geçirme raporu oluşturur:
  - İncelemeye katılan kişi sayısı
  - İnceleme süresi
  - Görüntülenen belgenin boyutu (örneğin, kod satırı sayısı, belgedeki sayfa sayısı)
  - Bulunan kusurların ciddiyetine göre ayrılmış sayısı (büyük/küçük)
- Bu gözden geçirme sürecinin bittiği yerdir.

3.2.3 Gözden Geçirmelerde Görev ve Sorumluluklar

Yönetici

- Gözden geçirmelerin planlanmasından sorumludur
- Gözden geçirmelerin yapılmasına karar verir
- Personeli belirler ve bir bütçe ve zaman çerçevesi belirler
- İncelemenin maliyet etkinliğini sürekli olarak izler
- Tatmin edici olmayan sonuçlar durumunda kontrol kararlarını uygular

Yazar

- Gözden geçirilmekte olan iş ürününü oluşturur
- Gözden geçirilmekte olan iş ürünündeki kusurları ortadan kaldırır (gerekirse)
- Gözden gerilecek materyallerin incelenmek üzere hazırlanmasından sorumludur
- Gerektiğinde, gözden geçirenlerin incelenmekte olan çalışma ürününün teknik açıklamalarını sunabilir.

3.2.3 Gözden Geçirmelerde Görev ve Sorumluluklar

Moderatör (Moderator veya Facilitator)

- Gözden geçirme toplantılarının sorunsuz bir şekilde yürütülmesini sağlar
- Farklı bakış açılarını uzlaştırmak gerekiyorsa arabuluculuk yapar
- İnceleme toplantısında güvenli bir karşılıklı güven ve saygı ortamının yaratılmasını sağlar

Raportör (Scribe veya Recorder)

- Bireysel incelemenin bir parçası olarak tespit edilen ve rapor edilen potansiyel anormallikleri toplar
- İnceleme toplantısı sırasında bulunan yeni potansiyel kusurların yanı sıra toplantıda alınan kararları (eğer gerçekleşirse) kaydeder

3.2.3 Gözden Geçirmelerde Görev ve Sorumluluklar

Gözden Geçiren (Reviewer)

- Gözden geçirilen iş ürünündeki potansiyel kusurları belirleyerek bir inceleme gerçekleştirir
- Konunun uzmanı olabilir, proje üzerinde çalışan bir kişi, iş ürünüyle ilgilenen bir paydaş ve/veya belirli teknik veya iş tecrübesine sahip bir kişi olabilir
- Farklı bakış açılarını temsil edebilir (örneğin, test uzmanının, geliştiricinin, kullanıcının, operatörün, iş analistinin, kullanılabilirlik uzmanının bakış açısı)
- Gözden geçirenler, incelenmekte olan çalışma ürününde sorunlar bulduklarından inceleme sürecinde önemli bir rol oynarlar ve bu genellikle incelemelerin ana amacıdır.
- Gözden geçirmeyi yapanlara, buldukları sorunları hazırlamaları ve raporlamaları için yeterli zaman verilmelidir.
- Gözden geçirenler yorumlarını yazara değil ürüne yöneltmelidir.

Gözden Geçirme Lideri(Review leader)

- Gözden geçirme sürecinin genel sorumluluğunu üstlenir
- Gözden geçirmeye kimin katılacağına karar verir.
- Gözden geçirmenin yerini ve tarihini belirler
- Gözden geçirme toplantılarının organize edilmesinden sorumludur

3.2.3 Gözden Geçirmelerde Görev ve Sorumluluklar

Roller ve kişiler

- Bazı gözden geçirme türlerinde, bir kişi birden fazla rolü üstlenebilir ve gözden geçirme türüne bağlı olarak her rolle ilişkili etkinlikler de farklılık gösterebilir.

- Ayrıca, gözden geçirme sürecine (ve özellikle kusurların, açık noktaların ve kararların kaydedilmesine) yardımcı olacak araçların ortaya çıkmasıyla birlikte, çoğu zaman bir katip görevlendirmeye gerek kalmaz.

- Herhangi bir gözden geçirme toplantısı planlanmıyorsa, bir yazıcıya da ihtiyaç duyulmayacaktır.

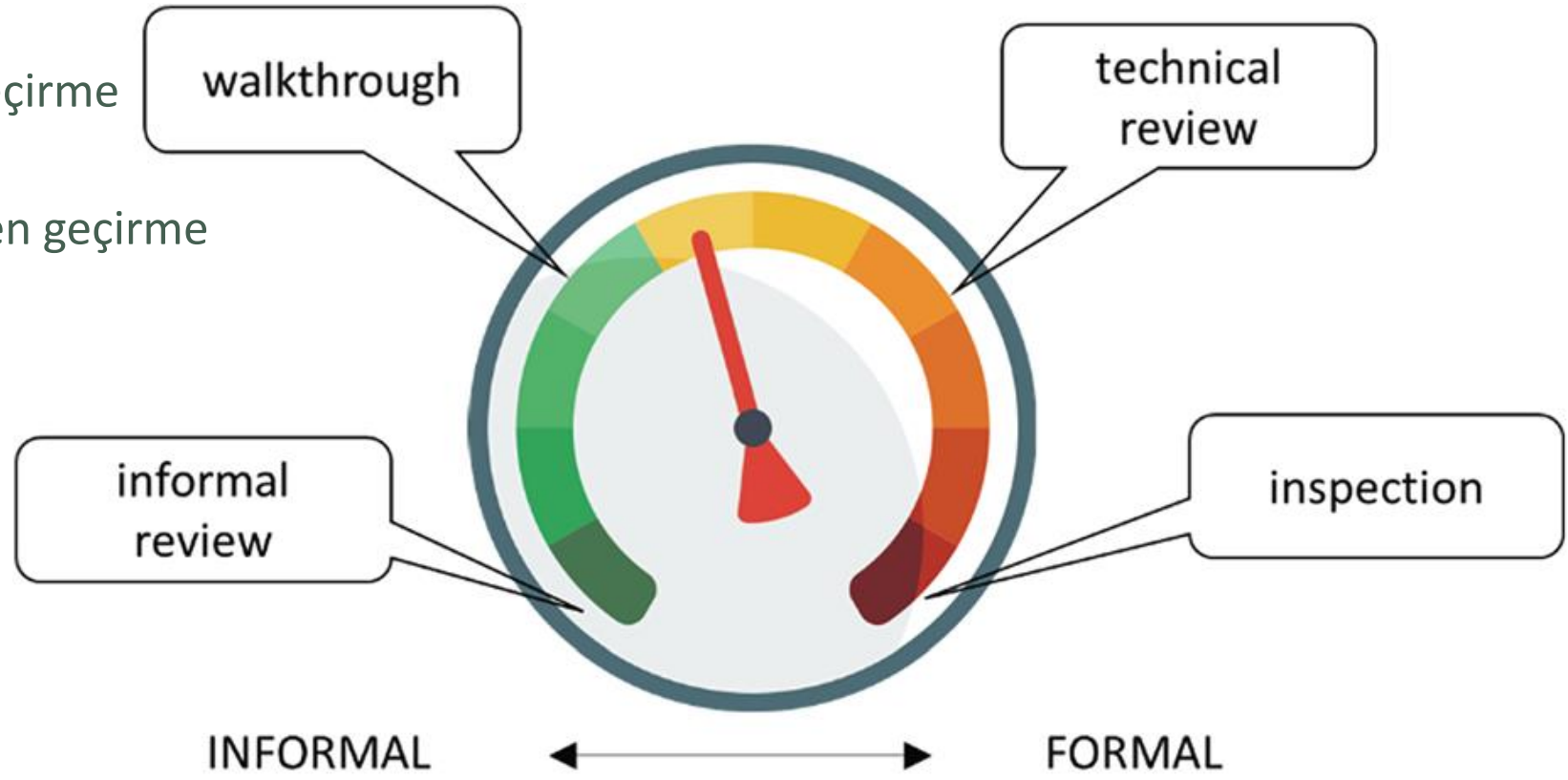
3.2.4 Gözden Geçirme Türleri

- Gayri resmi Gözden Geçirmelerden son derece resmileştirilmiş Gözden Geçirmelere kadar, farklı formalite düzeylerine sahip birçok Gözden Geçirme türü vardır.
- Gereken formalite düzeyi, kullanılan SDLC modeli, geliştirme sürecinin olgunluğu, incelenen iş ürününün kritikliği ve karmaşıklığı, yasal veya düzenleyici gereklilikler ve denetim takibi ihtiyacı gibi faktörlere bağlıdır.
- Doğru gözden geçirme türünü seçmek, gerekli hedeflere ulaşmak için kritik öneme sahiptir.
- Seçim yalnızca hedeflere değil aynı zamanda proje ihtiyaçları, mevcut kaynaklar, iş ürünü türü, riskler, iş alanı ve organizasyon kültürü gibi faktörlere de dayanmaktadır.

3.2.4 Gözden Geçirme Türleri

- Temel Düzey ders programı 4 tür gözden geçirmeyi tanımlar:

- Inspection – denetim
- Technical review – Teknik gözden geçirme
- Walkthrough – Üzerinden geçme
- Informal review – Gayrı resmi gözden geçirme



3.2.4 Gözden Geçirme Türleri

- Denetim resmi bir gözden geçirmedir.
- Teknik gözden geçirme ve üzerinden geçme, çok resmileştirilmişten oldukça gayri resmiye kadar değişebilir.
- Resmi olmayan gözden geçirmenin ayırt edici özelliği, tanımlanmış bir süreci takip etmemeleri ve bunlar aracılığıyla elde edilen bilgilerin resmi olarak belgelenmesine gerek olmamasıdır.
- Öte yandan, resmi gözden geçirmeler, belgelenmiş prosedürlere uygun olarak ve önceden oluşturulmuş bir ekibin katılımıyla yürütülür ve sonuçlar, zorunlu olarak belgelenmiştir.
- Gözden geçirmeler çeşitli amaçlarla yapılabilir, ancak en önemli amaçlardan biri de kusurları bulmaktır.
- Gözden geçirmeler çeşitli niteliklere göre sınıflandırılabilir.

3.2.4 Gözden Geçirme Türleri

Türü	Informal review	Walkthrough	Technical review	Inspection
Ana hedefler	Potansiyel kusurları tespit edin	Potansiyel kusurları tespit edin, kaliteyi artırın, alternatifleri değerlendirin, standartlara uygunluğu değerlendirin	Fikir birliği sağlayın, potansiyel kusurları tespit edin	Potansiyel kusurları tespit edin, iş ürününün kalitesini değerlendirin, ona olan güveni artırın, gelecekte benzer kusurların oluşmasını önleyin
Potansiyel ek hedefler	Yeni fikirler üretin, basit sorunları hızla çözün	Bilgi alışverişinde bulunun, katılımcıları eğitin, fikir birliğine varın	Bir çalışma ürününün kalitesini değerlendirmek, ona olan güveni artırmak, yeni fikirler üretmek, yazarları gelecekteki çalışma ürünlerini iyileştirme konusunda motive etmek, alternatifleri değerlendirmek	Yazarları gelecekteki çalışma ürünlerini ve yazılım geliştirme sürecini iyileştirmeye motive edin, bunun için koşullar yaratın, fikir birliğine varın
Resmi süreç	Hiçbiri	Toplantı öncesi isteğe bağlı bireysel hazırlık	Toplantı öncesi zorunlu bireysel hazırlık	Kurallara ve kontrol listelerine dayalı resmi süreç; Toplantıdan önce zorunlu bireysel hazırlık

3.2.4 Gözden Geçirme Türleri

Türü	Informal review	Walkthrough	Technical review	Inspection
Roller	Bir yazar, bir yazarın meslektaşı veya bir grup insan tarafından yürütülebilir.	Toplantıya genellikle yazar başkanlık eder; bir katibin varlığı gereklidir	Toplantı yazar tarafından değil, bir moderatör tarafından yürütülmelidir; bir katibin varlığı gereklidir	Kesin olarak tanımlanmış; toplantı yazar tarafından değil, moderatör tarafından yönetilir; yazar zorunludur; isteğe bağlı okuyucu rolü
Sonuçların belgelenmesi	İsteğe bağlı	İsteğe bağlı hata günlükleri ve inceleme raporları oluşturulur	Genel olarak kusur günlükleri ve inceleme raporları oluşturulur	Genel olarak kusur günlükleri ve inceleme raporları oluşturulur
Resmiyet Düzeyi	Gayri resmi	Gayri resmiden resmiye; senaryolar, provalar veya simülasyonlar şeklinde olabilir	Gayri resmiden resmiye; gözden geçirme toplantısı isteğe bağlı	Resmi; giriş ve çıkış kriterleri mevcuttur; Denetim süreci de dahil olmak üzere tüm süreci iyileştirmek için kullanılan önlemler toplanır
Kontrol listeleri	İsteğe bağlı	İsteğe bağlı	İsteğe bağlı	Genellikle kullanılır

3.2.4 Gözden Geçirme Türleri

Informal Review

- Gayri resmi gözden geçirme, yerleşik veya önceden tanımlanmış herhangi bir süreci takip etmez.
- Gayri resmi bir gözden geçirmenin sonucu da herhangi bir resmi şekilde belgelenmez.
- Gayri resmi bir gözden geçirmenin temel amacı potansiyel anormallikleri tespit etmektir.
- Gayri resmi gözden geçirmeye bir örnek, iki geliştirici arasında, birinin derleme hatasına neden olan bir kusuru bulmak için diğerinden yardım istemesi sırasında gerçekleşen bir konuşma olabilir.
- Başka bir örnek, ofis mutfağında öğle yemeği sırasında iki mühendis arasında bazı teknik sorunlar hakkında yapılan konuşma olabilir.
- Gayri resmi bir gözden geçirmeden sonra çoğu zaman hiçbir iz kalmaz ve hiçbir şey belgelenmez.
- Çevik metodolojilerde bu en yaygın inceleme türüdür.
- Gayri resmi gözden geçirme türlerine örnekler şunları içerir:
  - Buddy check - Arkadaş kontrolü
  - Pair review - Eş incelemesi

3.2.4 Gözden Geçirme Türleri

Walkthrough

- Walkthrough, bir çalışma ürününün sıralı olarak gözden geçirilmesini içerir.
- Çalışma ürününün yazarı tarafından gerçekleştirilir ve iş ürününün kalitesinin değerlendirilmesi, iş ürününe olan güvenin arttırılması, inceleyenlerin iş ürünü hakkındaki anlayışının geliştirilmesi, fikir birliğine varılması, yeni fikirler üretilmesi gibi birçok farklı hedefi içerebilir ve yazarları daha iyi ürünler oluşturmaya, kusurları daha etkili bir şekilde bulmaya motive eder.
- Gözden geçirenler, Walkthrough toplantısından önce bireysel hazırlık yapabilirler ancak bu zorunlu değildir.
- Bu inceleme türü, ekip bir yazılım arızasının nedenini tespit edemediğinde sıklıkla kullanılır.
- Ekip kodun nasıl çalıştığını iyi anlamak ve hataya neden olan kusuru bulmak için kodu satır satır dikkatlice analiz eder.
- Walkthrough kodun yazarı tarafından gerçekleştirilir, çünkü yazar genellikle kodun ne yaptığını (veya en azından kodu uygularken bu bağlamdaki niyetlerinin ne olduğunu) anında açıklayabilir.
- Kod incelemeleri gözden geçirme toplantılarında gerçekleştirildiğinde, bunlar genellikle bir Walkthrough biçimini alır.

3.2.4 Gözden Geçirme Türleri

Technical review

- Teknik açıdan nitelikli incelemeciler tarafından gerçekleştirilen Technical review'in amaçları, fikir birliği sağlamak ve teknik bir sorun hakkında karar vermek, aynı zamanda potansiyel kusurları tespit etmek, kaliteyi değerlendirmek ve iş ürününe güven oluşturmak, yeni fikirler üretmek ve motive etmek ve mümkün kılmaktır.
- Technical review genellikle bir "uzman paneli", yani yetkin kişilerin, projenin daha da geliştirilmesi üzerinde büyük etkisi olan bazı tasarım veya teknik kararları almakla görevlendirildiği bir toplantı şeklini alır.
- Bu nedenle toplantı öncesi bireysel hazırlık yapılması zorunludur, böylece toplantıya herkes bilgili ve yetkin bir şekilde katılabilir.
- Genel olarak Technical review toplantısı isteğe bağlıdır.
- Technical review genellikle bir raporla sona erer; özellikle de inceleme sırasında önemli tasarım kararları verilmişse geride bir iz kalması gerekir.

3.2.4 Gözden Geçirme Türleri

Inspection

- Inspection en resmi inceleme türlerinden biridir.
- Yazılım incelemelerinin gerçekleştirilmesi, getirdiği şüphesiz faydalar nedeniyle yazılım kalite mühendisliğinde en iyi uygulama olarak kabul edilir.
- Inspection, temel amacı kusur bulmada maksimum verimliliği sağlamaktır.
- Diğer hedefler kaliteyi değerlendirmek, çalışma ürününe güven oluşturmak ve yazarları çalışma ürünlerini iyileştirme konusunda motive etmek ve mümkün kılmaktır.
- Inspection özel bir özelliği, metrikleri toplamak ve bunları, denetim sürecinin kendisi de dahil olmak üzere tüm yazılım geliştirme sürecini iyileştirmek için kullanmaktır.
- Inspectionda yazar, inceleme lideri, moderatör veya yazar rolünü üstlenmemelidir.
- Inspection, kurallara ve kontrol listelerine dayalı, iyi tanımlanmış bir süreci takip eder.
- Inspection sonuçları belgelendirilmelidir.

3.2.5 Gözden Geçirmelere İlişkin Başarı Faktörleri

Başarılı bir gözden geçirmenin anahtarı, türünün ve kullanılan tekniklerin dikkatli seçilmesidir. Başarı faktörleri aşağıdaki faktörleri içerir ancak bunlarla sınırlı değildir:

- Her incelemenin, planlama sırasında tanımlanmış ve ölçülebilir çıkış kriteri olarak hizmet edebilecek açık hedefleri vardır.
- Kullanılan inceleme türleri, belirtilen hedeflere ulaşmaya yardımcı olur ve yazılım çalışması ürünlerinin türüne ve düzeyine ve katılımcılara uygundur.
- Bir iş ürününde mevcut olan kusurları etkili bir şekilde belirlemek için çeşitli inceleme teknikleri (kontrol listesi bazlı veya rol bazlı inceleme gibi) kullanılır.
- Kullanılan kontrol listeleri günceldir ve ana risklere yöneliktir.
- Büyük belgeler gruplar halinde yazılır ve gözden geçirilir, böylece yazarlar kusurlar hakkında hızlı ve sık geri bildirim alır (bu da kalite kontrolün bir parçasıdır).
- Katılımcıların incelemeye hazırlanmak için yeterli zamanı vardır; incelemeler önceden planlanır.
- Yazarlara, hakemlerden geri bildirim verilir, böylece onlar da çalışmalarının ürünlerini ve çalışmalarının kalitesini geliştirebilirler.
- Yönetim, inceleme sürecini destekler (örneğin, proje programında inceleme faaliyetleri için yeterli zaman ayırarak).
- İncelemeler, öğrenmeyi ve ürün ve süreç iyileştirmeyi teşvik etmek için organizasyon kültürünün doğal bir parçası olarak kabul edilir.

3.2.5 Gözden Geçirmelere İlişkin Başarı Faktörleri

- İnceleme, katılımı hedeflere ulaşmaya yardımcı olan kişileri içerir; örneğin, çalışmaları sırasında belgeyi potansiyel olarak kullanacak farklı becerilere veya bakış açılarına sahip kişiler.
- Test uzmanları, incelemenin önemli katılımcıları olarak kabul edilir ve çalışma ürünü hakkında kazandıkları bilgiler, önceden daha etkili testler hazırlamalarına olanak tanır.
- Katılımcılar incelemeye katılmak için yeterli zaman ayırırlar ve detaylara gereken özeni gösterirler.
- İncelemeler, bireysel inceleme ve/veya inceleme toplantısı (eğer yapılıyorsa) sırasında gözden geçirenlerin odak noktasını kaybetmemeleri için küçük parçalar üzerinde gerçekleştirilir.
- Tespit edilen kusurlar kabul edilir, onaylanır ve objektif bir şekilde ele alınır.
- Katılımcıların gereksiz faaliyetlerle zaman kaybetmemeleri için gözden geçirme toplantıları doğru şekilde yönetilir.
- İnceleme karşılıklı güven ortamında yapılır ve sonuçları katılımcıları değerlendirmek için kullanılmaz.
- Katılımcılar diğer katılımcılara karşı can sıkıntısı, kızgınlık veya düşmanlık belirtisi gösterebilecek jest ve davranışlardan kaçınırlar.
- Özellikle daha resmi inceleme türleri (teftişler gibi) için katılımcılara gerekli eğitim verildi.
- Bilginin genişletilmesine ve süreçlerin iyileştirilmesine olanak sağlayan bir atmosfer yaratılmıştır.

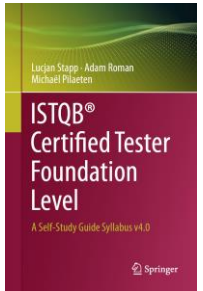
3.2.6 Gözden Geçirme Teknikleri

- Önceki versiyondaki (3.1) Temel Düzey ders programı, bir gözden geçirenin bireysel inceleme faaliyetinin, yani inceleme için bireysel hazırlığın bir parçası olarak kullanabileceği beş farklı tekniği tanımlıyordu.
- Bu tekniklerin amacı, gözden geçirilen iş ürünündeki kusurları tespit etmektir.
- Bahsedilen beş teknik şunlardır:
  - Özel amaçlı gözden geçirme
  - Kontrol listesi tabanlı gözden geçirme
  - Senaryolar ve provalar
  - Rol tabanlı gözden geçirme
  - Perspektif tabanlı okuma

Yazılım Test ve Doğrulama

Dr. Öğr. Üyesi Hakan KEKÜL
Sivas Cumhuriyet Üniversitesi
Teknoloji Fakültesi
Yazılım Mühendisliği

Bu ders notu genel olarak;
ISTQB® Certified Tester Foundation Level - A Self-StudyGuideSyllabusv4.0
Lucjan Stapp • Adam Roman • Michaël Pilaeten - Springer 2023
kitabındaki bilgilerden üretilmiştir.



Bölüm 4 Test Analizi ve Tasarımı

Anahtar Kelimeler

- **Acceptance criteria (Kabul kriterleri):** bir bileşenin veya sistemin bir kullanıcı, müşteri veya başka bir yetkili kuruluş tarafından kabul edilebilmesi için karşılaması gereken kriterlerdir.Referanslar: ISO 24765.
- **Acceptance test-driven development (Kabul testi odaklı geliştirme):** kabul testlerini paydaşların etki alanı dilinde tanımlayan işbirliğine dayalı ilk test yaklaşımıdır. Kısaltma: ATDD.
- **Black-box testing technique (Kara kutu test tekniği):** bir bileşenin veya sistemin spesifikasyonunun analizine dayanan bir test tekniğidir. Eşanlamlılar: black-box test design technique(kara kutu test tasarım tekniği), , specification-based technique (spesifikasyona dayalı teknik).
- **Boundary value analysis (Sınır değer analizi):** test senaryolarının sınır değerlerine göre tasarlandığı bir kara kutu test tekniğidir.
- **Branch coverage (dal kapsamı):** bir kontrol akış grafiğindeki dalların kapsamını oluşturur.
- **Checklist-based testing (Kontrol listesine dayalı test):** test senaryolarının bir kontrol listesinin öğelerini uygulamak üzere tasarlandığı deneyime dayalı bir test tekniğidir.
- **Collaboration-based test approach (İşbirliğine dayalı test yaklaşımı):** paydaşlar arasında işbirliği yaparak kusurlardan kaçınmaya odaklanan bir test yaklaşımıdır.
- **Coverage (Kapsam):** belirli kapsam öğelerinin bir test paketi tarafından yüzde olarak ifade edilme derecesidir. Eş anlamlı: test coverage (test kapsamı).
- **Coverage item (Kapsam öğesi):** bir test tekniği kullanılarak bir veya daha fazla test koşulundan elde edilen bir özellik veya niteliklerin birleşimidir.

Anahtar Kelimeler

- **Decision table testing (Karar tablosu testi):** Test senaryolarının, bir karar tablosunda gösterilen koşulların ve sonuçta ortaya çıkan eylemlerin kombinasyonlarını uygulamak üzere tasarlandığı bir kara kutu test tekniğini test eden karar tablosu.
- **Equivalence partitioning (Eşdeğerlik bölümlendirmesi):** test koşullarının her bölümün bir temsili üyesi tarafından uygulanan eşdeğerlik bölümleri olduğu bir kara kutu test tekniğidir. ISO 29119-1'den sonra. Eş anlamlı: partition testing (bölüm testi).
- **Error guessing(Hata tahmini):** Testlerin, test uzmanının geçmiş arızalara ilişkin bilgisine veya arıza modlarına ilişkin genel bilgisine dayanarak türetildiği bir test tekniğinde hata tahmininde bulunulması. Referanslar: ISO 29119-1.
- **Experience-based test technique (Deneyime dayalı test tekniği):** testi yapan kişinin deneyimine, bilgisine ve sezgisine dayanan bir test tekniğidir. Eş anlamlı : experience-based test design technique, experience-based technique.
- **Exploratory testing (Keşifsel test):** test uzmanlarının bilgilerine, test öğesinin araştırılmasına ve önceki testlerin sonuçlarına dayanarak testleri dinamik olarak tasarlayıp yürüttüğü bir test yaklaşımıdır. ISO 29119-1'den sonra.
- **State transition testing (Durum geçiş testi):** test senaryolarının bir durum geçiş modelinin öğelerini uygulamak üzere tasarlandığı bir kara kutu test tekniğidir. Eş anlamlı : finite state testing.
- **Statement coverage:** yürütülebilir ifadelerin kapsamı.
- **Test technique (Test tekniği):** test koşullarını tanımlamak, test senaryolarını tasarlamak ve test verilerini belirlemek için kullanılan bir prosedürdür. Eş anlamlılar: test design technique (test tasarım tekniği).
- White-box test technique (Beyaz kutu test tekniği): yalnızca bir bileşenin veya sistemin iç yapısına dayanan bir test tekniğidir. Eş anlamlı: white-box test design technique, structure-based technique.

4.1 Test Tekniklerine Genel Bakış

FL-4.1.1 (K2) Kara kutu, beyaz kutu ve deneyime dayalı test tekniklerini ayırt edin.

4.1 Test Tekniklerine Genel Bakış

- Bu bölümde şunları açıklıyoruz:
  - Doğru test tekniği nasıl seçilir
  - Temel Düzey ders programında ne tür test teknikleri anlatılmaktadır
  - Her bir test tekniği grubunun ortak özellikleri nelerdir

- Yazılım testi bağlamında bu aktivite, yazılım bilgisinin bilimsel temeline dayalı olarak test koşullarını, test senaryolarını ve test verilerini türetmek (üretmek) için belirli yöntemlerin kullanılmasını içerecektir.

- Temel Düzey ders programı dokuz test tekniğini tartışıyor ve bunları üç kategoriye ayırıyor:
  - Black-box Test Techniques - Kara kutu test teknikleri (dört teknik)
  - White-Box Testing Techniques - Beyaz kutu test teknikleri (iki teknik)
  - Experience-Based Testing Techniques - Deneyime dayalı test teknikleri (üç teknik)

4.1 Test Tekniklerine Genel Bakış

Kara Kutu Test Teknikleri

- Kara kutu test teknikleri aynı zamanda davranışsal teknikler veya spesifikasyona dayalı teknikler olarak da bilinir.
- Nasıl davranması gerektiği konusunda test nesnesinin dışındaki bilgileri kullanırlar.
- Bu bilgi, örneğin gereksinimlerin spesifikasyonu, kullanım senaryosu açıklaması, kullanıcı hikayeleri (çevik projelerde) ve iş süreçleri olabilir.
- Bu modellerin tümü, iç tasarımına atıfta bulunmadan, sistemin hem işlevsel hem de işlevsel olmayan istenen davranışını tanımlar.
- Kara kutu tekniklerini kullanmanın avantajı, yukarıda bahsedilen belgelerin genellikle bir bileşenin veya sistemin uygulanması başlamadan çok önce mevcut olmasıdır.
- Bu, test aktivitelerinin (örn. test analizi ve test tasarımı) kod geliştirmeden çok önce başlayabileceği anlamına gelir.
- Bu, kara kutu test tekniklerini yalnızca dinamik test sırasında değil, aynı zamanda test tasarımı aşamasında da bir tür statik test yöntemi olarak kullanmamıza olanak tanır.
- Kara kutu teknikleri hem işlevsel hem de işlevsel olmayan testlerde kullanılabilir.

4.1 Test Tekniklerine Genel Bakış

Beyaz Kutu Test Teknikleri

- Beyaz kutu test tekniklerine aynı zamanda yapılandırılmış veya yapı tabanlı teknikler de denir.
- Beyaz kutu test tasarımının temeli, test nesnesinin iç yapısıdır.
- Çoğu zaman bu yapı kaynak kodudur, ancak aynı zamanda sistemin veya modül mimarisinin daha yüksek düzeydeki başka bir modeli de olabilir. Örneğin, entegrasyon testi düzeyinde böyle bir model, çağrı grafiği olarak adlandırılabilir ve sistem testi düzeyinde, iş sürecindeki bilgi akışı olabilir.
- Programın iç yapısına ilişkin bilgi, yalnızca bu modelin belirli öğelerini (örneğin kod ifadeleri, koddaki kararlar, programdaki yollar) kapsayan testleri tasarlamak için kullanılır.
- Buna karşılık, her test için beklenen çıktı, test edilen sistemin dışındaki bilgilere, örneğin spesifikasyonlara veya sağduyuya dayanarak belirlenmelidir.

4.1 Test Tekniklerine Genel Bakış

Deneyime Dayalı Test Teknikleri

- Deneyime dayalı test teknikleri grubu, herhangi bir resmi belgeye (tasarıma, gereksinimlere, koda vb.) dayanmaması açısından diğer iki gruptan farklılık göstermektedir.
- Bunun nedeni, deneyime dayalı test tekniklerinin, test edilen sistem hakkında biraz daha "yumuşak" bilgi kaynaklarını kullanmasıdır.
- Bu kaynaklar bilgi, sezgi, deneyim, sistemin önceki sürümlerinde veya benzer uygulamalarda bulunan kusurlara ilişkin bilgi vb.'dir.
- Bu nedenle, test edilen sistem hakkında "nesnel" bilgiden ziyade, doğrudan test uzmanlarının nitelik ve becerilerine atıfta bulunurlar.
- Teknikler yalnızca test uzmanlarının değil aynı zamanda diğer tüm proje paydaşlarının deneyimlerinden de faydalanabilir.
- Örnekler arasında geliştiricilerin, son kullanıcıların, müşterilerin, mimarların, iş analistlerinin ve proje yöneticisinin uzmanlığının kullanılması yer alır.
- Deneyime dayalı test teknikleri sıklıkla kara kutu ve beyaz kutu test teknikleriyle birlikte kullanılır.
- Bu, test uzmanlarına, projenin tüm nüanslarını veya yazılımın geliştirildiği bağlamı hesaba katamayan, daha resmi test teknikleri kullanılarak kolayca gözden kaçan sorunları tespit etme şansı verir.

4.1 Test Tekniklerine Genel Bakış

Karşılaştırma kriteri	Black-box	White-box	Experience-based
Test koşullarının, test senaryolarının ve test verilerinin türetilmesi için kaynak	Test nesnesinin dışındaki test esası: gereksinimler, özellikler, kullanım senaryoları, kullanıcı hikayeleri	Test nesnesinin iç yapısını açıklayan test esası: kod, mimari, ayrıntılı tasarım; Spesifikasyonlar genellikle beklenen sonuçları tanımlamak için kullanılır.	Test uzmanlarının, geliştiricilerin, kullanıcıların ve diğer paydaşların bilgi, deneyim ve sezgileri
Tespit edilen sorunların türü	Gereksinimler (beyan edilen davranış) ile bunların uygulanması arasındaki tutarsızlıklar	Kontrol akışı veya veri akışıyla ilgili sorunlar	Testleri gerçekleştiren kişiye veya örneğin testlerde kullanılan kontrol listesine bağlıdır
Kapsam	Test esasının test edilen unsurlarına ve test esasına uygulanan tekniğe göre ölçülür	Kod veya arayüzler gibi yapının test edilen öğelerine göre ölçülmüştür	Tanımlanmamış

4.1 Test Tekniklerine Genel Bakış

TEST TEKNİKLERİ

Black – Box

- Equivalence Partitioning (EP – Eş Değer Aralık)
- Boundary Value Analysis (BVA - Sınır Değer Analizi)
- Decision Table Testing (Karar Tablosu Testi)
- State Transition Testing (Durum Geçiş Testi)

White – Box

- Statement Testing and Statement Coverage (İfade Testi ve Kapsamı)
- Branch Testing and Branch Coverage (Dal Testi ve Kapsamı)

Experince – Based

- Error Guessing (Hata Tahmini)
- Exploratory Testing (Araştırma testi)
- Checklist-Based Testing (Kontrol Listesi Tabanlı Test)

4.1 Test Tekniklerine Genel Bakış

Test Tekniğinin Seçilmesi

- Test tekniklerinin seçimini birçok faktör etkiler.
- Bunlar üç ana gruba ayrılabilir:
  - Biçimsel faktörler (örneğin, belgeler, yürürlükteki yasa ve yönetmelikler, müşteri sözleşmesi hükümleri, kuruluşta mevcut süreçler, test hedefleri, kullanılan SDLC modeli).
  - Ürün faktörleri (örneğin yazılım, karmaşıklığı, çeşitli kalite özelliklerinin önemi, riskler, beklenen kusur türleri, yazılımın beklenen kullanımı).
  - Proje faktörleri (örn. mevcut zaman, bütçe, kaynaklar, araçlar, beceriler, bilgi ve test uzmanlarının deneyimi).

4.1 Test Tekniklerine Genel Bakış

Test Tekniğinin Seçilmesi

Örnek

- Bir üniversite kütüphanesi uygulamasının geliştirilmesini içeren bir proje üzerinde çalışıyorsunuz.
- Kuruluşunuzda, geliştirilmekte olan uygulamaları test etme zorunluluğu bulunurken, projenizde uygulanan test stratejisi, müşteriyle imzalanan sözleşmeler nedeniyle resmi test kullanarak test yapma ihtiyacını zorunlu kılar.
- Proje V modeli altında yürütülmektedir.
- Gereksinimler, gereksinim spesifikasyonu şeklinde toplanmış ve sunulmuştur.
- Sistemin mimarları şu anda bir kullanıcının bu kullanıcı tipine (öğrenci, çalışan) ve gecikme cezalarına göre kaç kitap ödünç alabileceğinin doğrulanmasından sorumlu modülün iş mantığının tasarımı üzerinde çalışıyor.
- Bu, kütüphanenin kural ve düzenlemelerinin doğru şekilde uygulanmasını sağlamak için önemli bir bileşendir, dolayısıyla arıza riskinin etkisi çok yüksektir.
- Manuel testleri analiz etmek ve tasarlamak için 3 gününüz var.
- Regresyon testinin bu projede büyük bir rol oynamayacağından dolayı test otomasyonu beklenmemektedir.
- Ayrıca kuruluş, otomatik test komut dosyaları oluşturmaya yönelik özel araçlara veya teknik becerilere sahip değildir.

4.1 Test Tekniklerine Genel Bakış

Test Tekniğinin Seçilmesi

Örnek

- Yukarıdaki açıklamadan, kitap ödünç verme kurallarının uygulanmasından sorumlu bileşen bağlamında aşağıdakiler açıkça görülmektedir:
  - Testler yapmamız gerekiyor (bu doğrudan belgelerden ve sözleşmeden gelir).
  - Bu testleri tasarlamamız ve belgelememiz gerekiyor.
  - Test iş mantığına odaklanmalıdır, dolayısıyla yazılımın bu yönü için uygun test tekniklerinin kullanılması mantıklı bir seçim olacaktır.
  - Bu bileşenle ilişkili risk etkisi yüksektir, dolayısıyla iş mantığını çok dikkatli bir şekilde kontrol eden yöntemlerin kullanılması mantıklıdır.
  - Test manuel olarak yapılacaktır (araç ve deneyim eksikliği, genellikle otomasyona doğal bir aday olan regresyon testinin düşük rolü).

4.1 Test Tekniklerine Genel Bakış

Test Tekniğinin Seçilmesi
Örnek

- Yukarıdaki örnekten görülebileceği gibi, tekniğin seçimine ilişkin karar, analizin ayrıntı düzeyi, test tasarımının seçimi, uygulama ve yürütme birçok biçimsel, ürün ve proje faktöründen etkilenir.
- Mükemmel, evrensel bir test tekniği yoktur.
- Her tekniğin kendine göre avantaj ve dezavantajları bulunmaktadır.
- Her biri bir durumda daha iyi, diğerinde ise daha kötü performans gösterecektir.
- Örneğin, bir kütüphanede kitap ödünç verme kurallarını doğrulayan bir bileşenle ilgili yukarıda bahsedilen durum söz konusu olduğunda, kara kutu test tekniklerinden biri olan karar tablosu testi iyi bir seçim gibi görünüyor çünkü iş mantığını test ediyor ve bu, test etmeye en çok önem verdiğimiz işlevselliktir.
- Öte yandan, örneğin programın iç yapısına (kaynak kodu) dayanan beyaz kutu test tekniklerini kullanmanın bir anlamı yoktur, çünkü buradaki test sorunu iş mantığı testidir.
- Deneyimli test uzmanları tarafından sıklıkla kullanılan iyi bir uygulama, teknikleri birleştirmektir.
- Örneğin, yukarıdaki problemde karar tablolarını kullanarak, testi yapan kişi zaman zaman sınır değer analizi tekniğini uygulayabilir ve bu tür değerler için iş kurallarını kontrol edebilir.

4.1 Test Tekniklerine Genel Bakış

Test Tekniğinin Seçilmesi
Örnek

- Test tekniklerinin kullanımı aynı zamanda test seviyeleri bağlamında da düşünülebilir.
- Bazı teknikler daha evrenseldir; dolayısıyla bileşen testinden kabul testine kadar tüm test seviyelerinde doğal olarak bulunurlar.
- Öte yandan diğer bazı tekniklerin uygulama kapsamı biraz daha sınırlıdır; örneğin, beyaz kutu teknikleri çoğunlukla bileşen testi düzeyinde uygulanır (örneğin, geliştirici birim testleri).
- Elbette bu tekniğin kabul testi düzeyinde kullanılmasına ilişkin örnekler hayal edilebilir, ancak pratikte bu tür durumlar oldukça nadirdir.
- Test teknikleri kullanılarak test geliştirmenin resmileştirilme derecesi, çok resmi olmayandan çok resmiye kadar değişebilir. Tablo 4.2'de değişen formalizasyon derecelerine ilişkin bazı örnekler veriyoruz.

4.2 Kara Kutu Test Teknikleri

FL-4.2.1 (K3) Test senaryolarını türetmek için eşdeğerlik bölümlendirmesini kullanın.
FL-4.2.2 (K3) Test senaryolarını türetmek için sınır değer analizini kullanın.
FL-4.2.3 (K3) Test senaryolarını türetmek için karar tablosu testini kullanın.
FL-4.2.4 (K3) Test senaryolarını türetmek için durum geçiş testini kullanın.

4.2 Kara Kutu Test Teknikleri

- Bu bölümde, ders programında açıklanan dört kara kutu test tekniğini tartışacağız:

- Use Case/Case based testing (Kullanım Senaryosu Testi/Vaka bazlı test). Eski sınav müfredatında olmasına rağmen güncel müfredat da mevcut değildir.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

- Yedi test ilkesinden biri "kapsamlı testlerin imkansız olduğunu" söylüyor.
- Bu oldukça açıktır: Giriş verilerinin olası kombinasyonlarının sayısı neredeyse sonsuzdur, oysa test uzmanı yalnızca sonlu ve çok küçük sayıda test gerçekleştirme yeteneğine sahiptir.
- Eş değer aralık tekniği, kapsamlı testlerin imkansızlığı ilkesinin üstesinden gelmeye çalışır.
- Genellikle sonsuz sayıda olası girdi vardır, ancak bir programın bu girdiler üzerindeki beklenen çeşitli davranışlarının sayısı, özellikle uygulamanın işleminin kesin olarak tanımlanmış bir yönü ile ilgili belirli davranışları dikkate aldığımızda, genellikle sınırlıdır.
- Bazı örneklere bakalım.

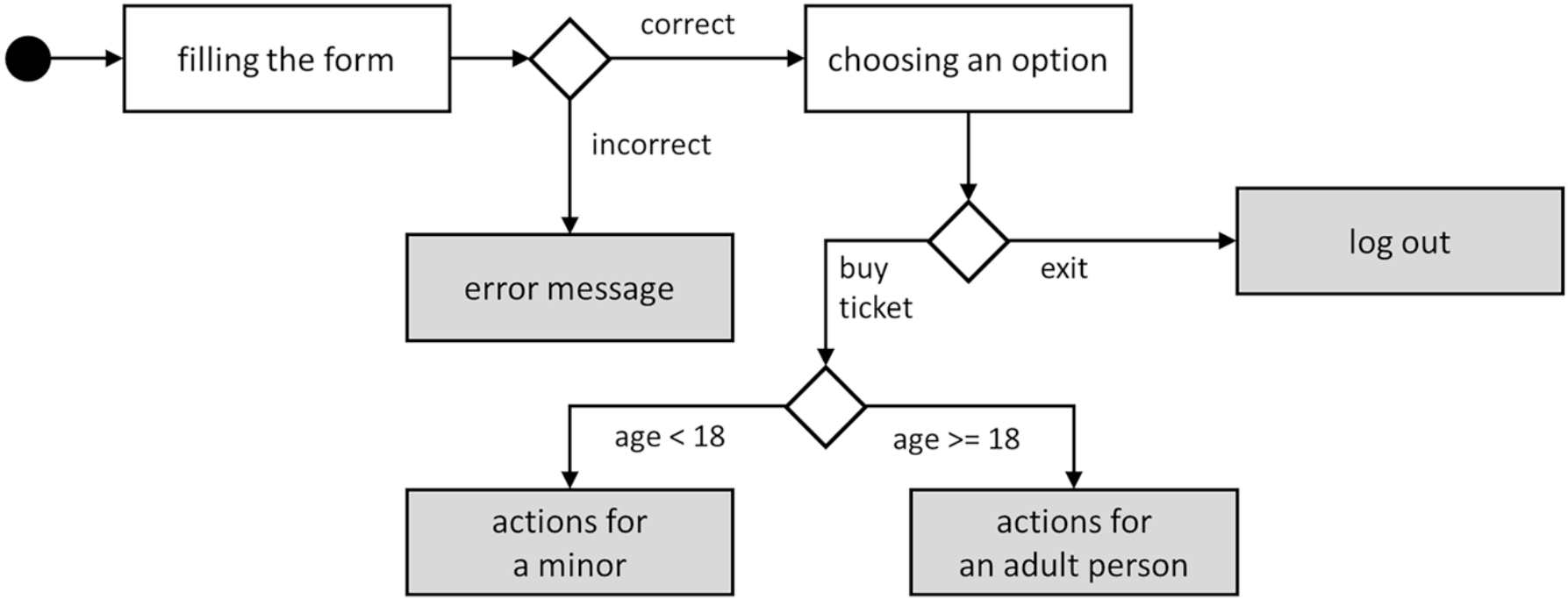
4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

- Örnek: Sistem vergi miktarını gelire göre belirliyor. Vergi %0, %19, %33 veya %45 olabilir. Gelir için potansiyel olarak sonsuz sayıda olası değer vardır, ancak sistem tarafından verilen yalnızca dört olası karar türü vardır.

- Örnek: Bir kullanıcı bir web formunu doldurur ve ardından yazdırmak ister. Çıktı bir veya iki taraflı olabilir. Bir formu doldurmanın potansiyel olarak sonsuz yolu vardır, özellikle de karmaşıksa. Ancak bu durumda test etmek istediğimiz yalnızca iki tür davranıştır: tek taraflı ve iki taraflı yazdırmayı düzeltin.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

- Örnek: Kullanıcı “yaş” alanı da dahil olmak üzere formu doldurur. Kullanıcı daha sonra, kullanıcının yaşına bağlı olarak farklı prosedürler uygulayarak bir bilet satın alabilir. Sürecin tam akış şeması Şekil 4.3'te gösterilmektedir. Formu doldurmak için pek çok olasılık vardır ancak yalnızca dört olası eylem vardır: hata mesajı, bir yetişkin tarafından bilet satın alma, reşit olmayan bir kişi tarafından bilet satın alma ve oturumu kapatma.



4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

- Gördüğünüz gibi, bir programın test davranışını sınırlı sayıda değişkene indirmek genellikle mümkündür.
- Eş değer aralık yöntemi, belirli bir etki alanını bölümler (veya eşdeğerlik bölümleri) adı verilen alt kümelere böler, böylece bir bölümdeki her iki öğe için aynı program davranışına sahip oluruz.
- Örneğin, sistem 18 yaşın altındaki kişilere bir indirim atarsa, 18'den küçük tüm sayılar, indirimin tahsisine karşılık gelen bir denklik bölümü oluşturacaktır.
- Bu sayede testçinin bakış açısına göre aynı bölüme ait değerler aynı şekilde ele alınır. Bu nedenle, belirli bir bölümün her öğesi test etmek için eşit derecede iyi bir seçimdir.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

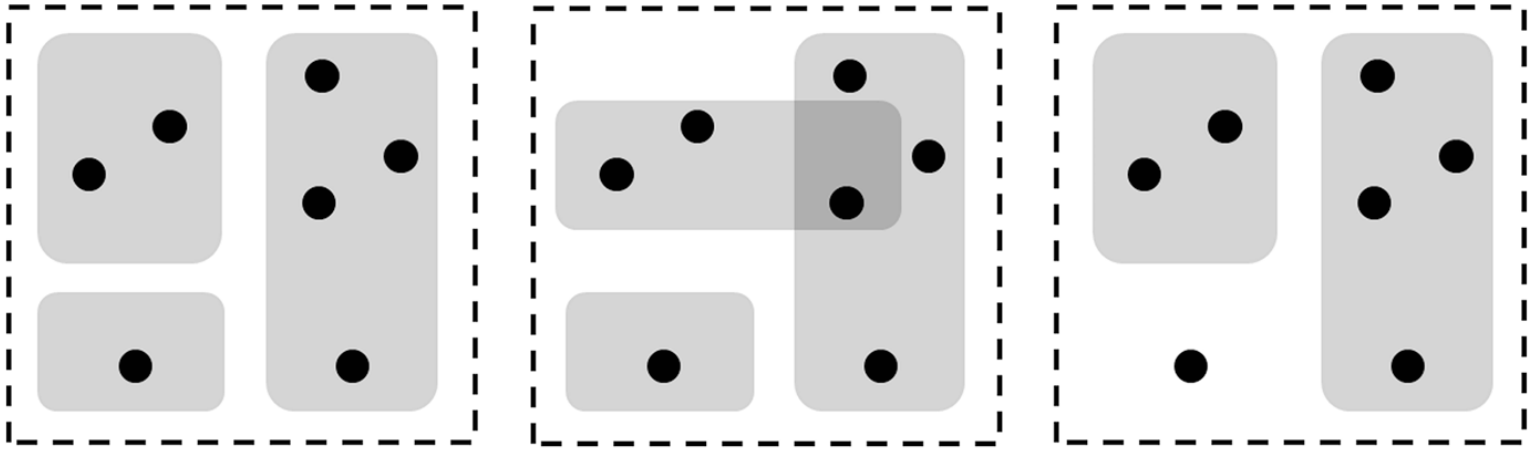
Uygulama

- EP tekniği çok yönlüdür.
- Hemen hemen her durumda, her test seviyesinde ve her test türünde kullanılabilir.
- Bunun nedeni, olası verilerin gruplara bölünmesiyle ilgili olmasıdır.
- Tekniğin yalnızca girdi alanlarına değil, aynı zamanda çıktı alanlarına ve iç alanlara da uygulanabileceğini belirtmekte fayda var.
- Eşdeğerlik bölümlerine ayırdığımız alanın sayısal bir alan olması gerekmez.
- Aslında boş olmayan herhangi bir küme olabilir.
- Eşdeğerlik bölümleme tekniğinin uygulanabileceği bazı alan örnekleri:
  - Bir dizi doğal sayı (örneğin, çift ve tek sayılara bölme)
  - Bir kelime koleksiyonu (örneğin, kelime uzunluğuna göre bölme, bir harf, iki harf vb.)
  - Zamanla ilgili koleksiyonlar (örneğin, doğum yılı, belirli bir yıldaki aya göre vb.)
  - İşletim sistemi türlerinden oluşan bir koleksiyon: {Windows, Linux, macOS}

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümleme doğruluğu

- Yaptığımız bölümlemenin doğru olması çok önemlidir, bu şu anlama gelir:
  - Alanın her öğesi tam olarak bir eş değer aralığa aittir.
  - Hiçbir eş değer aralık boş değildir



Doğru (a)
Hatalı (b, c)

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümleme doğruluğu

- Eş değer bölme meselesi oldukça açık görünüyor, ancak pratikte bu önemsiz bir sorun değil.
- Gerçek uygulamalarda etki alanları ve bölümler çok karmaşık bir yapıya sahip olabilir.
- Yaptığımız bir bölümlemede çoğu zaman bazı doğruluk koşullarının ihlal edildiği görülür.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümleme doğruluğu

- Örnek: klasik üçgen test problemini düşünün.
- Program, girdi olarak negatif olmayan üç tamsayıyı (a, b, c) alır ve verilen uzunluktaki parçalardan oluşturulabilecek üçgen tipini çıktı olarak verir.
- Programın olası cevapları “eşkenar üçgen”, “ikizkenar üçgen”, “çeşitkenar üçgen” ve “üçgen değil”dir.
- EP tekniğini
  - giriş alanına (yani negatif olmayan tamsayıların tüm olası üçlüleri (a, b, c) kümesine)
  - çıkış alanına (yani üçgenin türüne) uygulamak istediğimizi varsayalım.
- Giriş alanını daha önce sözü edilen dört olası çıktıya karşılık gelen dört bölüme bölmek doğal görünmektedir.
- Bununla birlikte, daha yakından incelendiğinde, her eşkenar üçgenin aynı zamanda bir ikizkenar üçgen olduğu ortaya çıkar; bu nedenle, başka bir bölümün uygun bir alt kümesi olan bir eşdeğerlik bölümüyle karşı karşıyayız.
- Aşağıdaki bölümleri tanımlayarak bölümlememizi düzeltmemiz gerekiyor:
  - Eşkenar üçgenleri temsil eden girdiler
  - Eşkenar olmayan ikizkenar üçgenleri temsil eden girdiler
  - Çeşitkenar üçgenleri temsil eden girdiler
  - “Üçgen değil” kategorisine giren girdiler
- Şimdi, bu bölümleme doğrudur: girdiye verilen sayıların her üçlüsü tam olarak bire karşılık gelir yukarıdaki dört olası bölümden.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümleme doğruluğu

- Örnek: Tüm olası sonlu sayı dizilerini sıralamalarına göre sınıflandırmak istiyoruz.
- Bu tür verilerin eşdeğerlik bölümlerine doğal bir eşdeğerlik bölümü şu şekilde görünebilir: artan diziler, azalan diziler ve sırasız diziler.
- Ancak tek elemanlı bir dizinin hem artan hem de azalan bir dizi olduğunu unutmayın.
- Ek olarak, boş dizenin (0 öğe içeren) nerede sınıflandırılacağı sorusu ortaya çıkar.
- Bu nedenle doğru bölümlemede bu "uç durumlar" dikkate alınmalıdır ve bölümler şu şekilde görünebilir:

    - Bölüm 1: boş dizi
    - Bölüm 2: tüm tek elemanlı diziler
    - Bölüm 3: en az iki öğeli tüm artan diziler
    - Bölüm 4: en az iki öğeli tüm azalan diziler
    - Bölüm 5: ikiden fazla öğeli tüm sırasız diziler

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümleme doğruluğu

- "Normal", "doğru" değerleri yani sistem tarafından beklenen (kabul edilen) değerleri içeren bölümlere **valid partitions - geçerli bölümler** denir.
- Bileşenin veya sistemin reddetmesi gereken değerleri (örneğin, yanlış söz dizimine sahip veriler, kabul edilebilir aralıkları aşan veriler vb.) içeren bölümlere **invalid partitions - geçersiz bölümler** denir.
- Son örneğimizde, tanımladığımız beş bölüm geçerlidir çünkü her biri sistem tarafından beklenen doğru verileri içerir (belki de boş dize durumu tartışmalıdır; eğer belirtim boş olmayan dizelerden açıkça bahsetmiyorsa, o zaman bölümleme 1 geçersiz bir bölüm olarak kabul edilebilir).
- Tanımlanan bu bölümlere ek olarak, örneğin sayısal dizeler olmayan öğelerden oluşan (örneğin alfasayısal karakterler içeren) geçersiz bir bölümü de ayırt edebiliriz.
- Geçerli ve geçersiz bölümlerin ve dolayısıyla geçerli ve geçersiz değerlerin tanımları farklı şekillerde anlaşılabilir, dolayısıyla bu terimleri kullanacaksak bunları tam olarak tanımlamakta fayda var.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümleme doğruluğu

- Örneğin geçerli değerler en az iki şekilde anlaşılabilir:
  - Sistem tarafından işlenmesi gerekenler olarak
  - Şartnamede işlenmeleri tanımlananlar olarak
- Benzer şekilde, yanlış değerler anlaşılabilir:
  - Sistem tarafından göz ardı edilmesi veya reddedilmesi gerekenler olarak
  - Şartnamede işlenmeleri tanımlanmayanlar olarak

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümleme doğruluğu

- Örnek: Kullanıcı kayıt formunda sistem “e-posta” alanına geçerli bir e-posta adresinin girilmesini beklemektedir.
- Kullanıcı buraya “abc@def@ghi” karakter dizisini girdiğinde sistem bu girişi geçersiz olarak reddederek kullanıcıya “Yanlış e-posta” mesajı ile bilgi verir.
- Bu durumda “abc@def@ghi” girişi
  - bazıları tarafından geçersiz bir değer olarak değerlendirilir (Sistem reddettiği için kayıt yalnızca doğru e-posta adresi girildiğinde gerçekleşecektir)
  - bazıları tarafından ise geçerli bölümden gelen bir değer olarak değerlendirilebilir (sistem bu tür durumlara hazır olduğundan bunun kanıtı hata mesajı; yani bir anlamda "beklenen" bir değerdir ve dolayısıyla geçerli bölümden gelir).

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümlerin Alt Bölümlere Daha Fazla Ayrılması

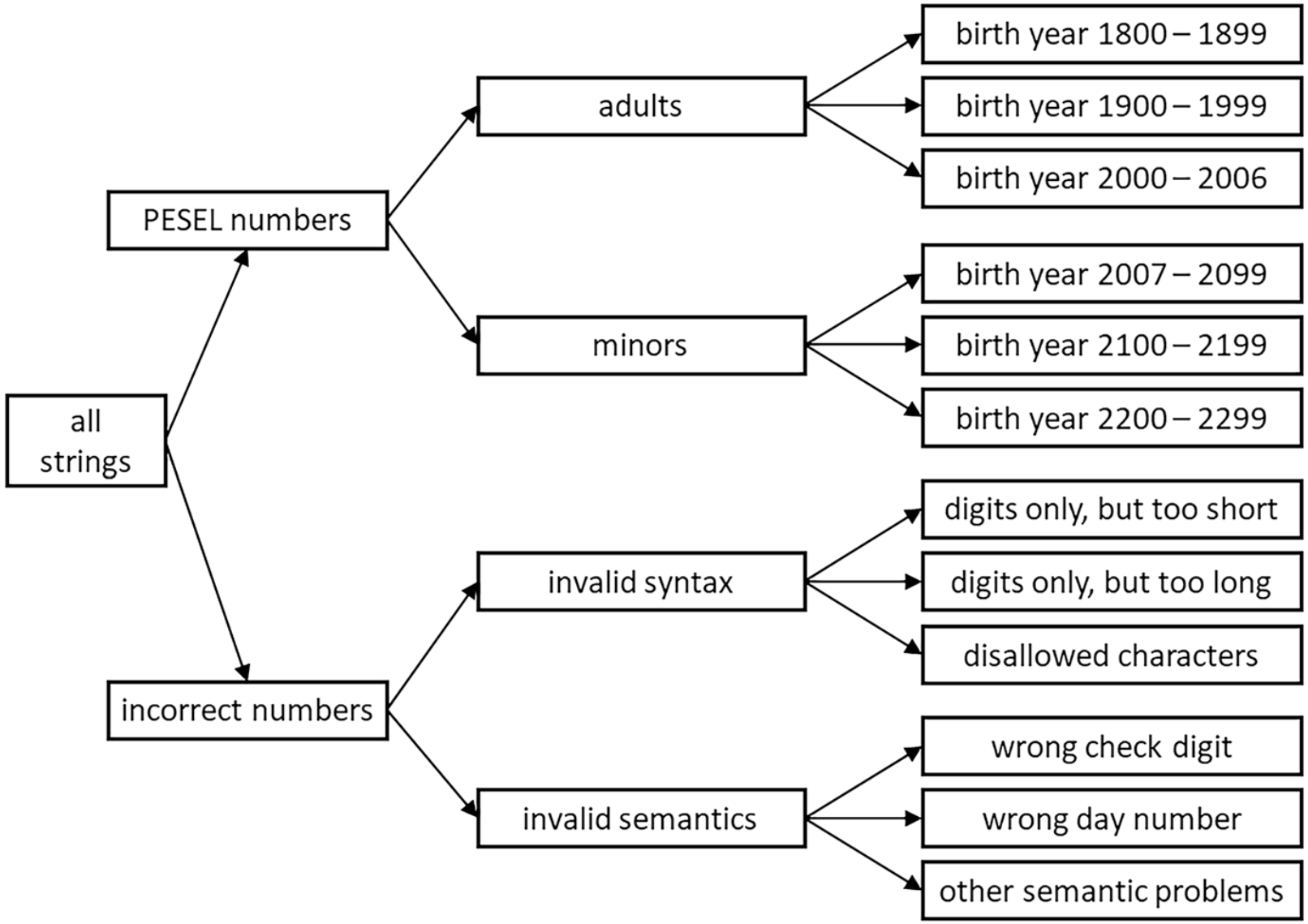
- EP tekniğinin önemli bir avantajı, herhangi bir bölümü (hatta tümünü) alt bölümlere ayırırsanız, yine eşdeğer bölümlemeye sahip olmanızdır.
- Böyle daha parçalı bir alan için test yapmak daha doğru olacaktır.
- Testi yapan kişi, belirli nedenlerden dolayı belirli bölümlerin daha da alt bölümlere ayrılması gerektiğine karar verebilir.
- Bu, bölümlerin hiyerarşik yapısından yararlanan doğal bir yaklaşımdır.
- Aşağıdaki örneği düşünün.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Bölümlerin Alt Bölümlere Daha Fazla Ayrılması

- Örnek: PESEL, Polonya hükümeti tarafından kullanılan kişisel vatandaş kimlik numarasıdır.
- Her Polonya vatandaşına tahsis edilmiş benzersiz bir PESEL numarası vardır.
- Bu, 11 haneli bir sayıdır; ilk altı rakamı doğum tarihini, son rakamı kontrol rakamını ve sondan bir önceki rakamın eşitliği cinsiyeti (erkek veya kadın) kodlar.
- Sistem kullanıcıdan PESEL numarasını alıyor ve kişinin reşit olup olmamasına göre gerekli işlemi yapıyor.
- Mümkün olan tüm rakam dizileri için denklik bölümlemesi yapmak istiyoruz.
- İlk adım olarak bunları geçerli ve geçersiz sayılara, yani geçerli PESEL numaralarını ve diğer her şeyi temsil eden sayılara ayırabiliriz.
- Daha sonra, bu bölümlerin her birini daha sonraki olası bölünme açısından ele alabiliriz.
- Doğru PESEL sayıları, spesifikasyona göre şüphesiz yetişkinlere ve küçüklere karşılık gelen sayılara bölünmelidir.
- Bu bölümlerin her birinin daha ileri bir bölümü, örneğin ay numarasının uygun kodlamasını hesaba katabilir
- Buna karşılık, PESEL numarası yapısal (örn. yanlış uzunluk) veya esasa ilişkin (örn. kontrol numarası tutarsızlığı) gibi çeşitli nedenlerden dolayı yanlış olabilir.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)



Bölme konusunda daha ileri gidebileceğimize dikkat edin. Örneğin, her geçerli denklik bölümü "erkek" ve "kadın" cinsiyetlerini kodlayan iki bölüme ayrılabilir.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Test Durumlarının Türetilmesi

- EP tekniği durumunda kapsam öğeleri eşdeğerlik bölümleridir.
- %100 kapsamı sağlamak için minimum test senaryoları seti, her eşdeğerlik bölümünü kapsayan bir test senaryosudur; yani tanımlanan her bölüm için, o bölümden bir değer içeren bir test senaryosu vardır.
- Tek boyutlu durumda (bir alan ve bir bölüm), minimum test senaryosu sayısı, tanımladığımız eşdeğerlik bölümleri kadar olmalıdır.
- Çok boyutlu durumda (birden fazla alan), test senaryolarının sayısı, örneğin geçersiz bölüm kombinasyonlarını ele alma şeklimize ve ayrıca olası bağımlılıklara veya kısıtlamalara bağlı olacağından, konu biraz daha karmaşık hale gelir.
- Kapsam, en az bir değer kullanılarak test edilen eşdeğerlik bölümlerinin sayısının tanımlanan toplam eşdeğerlik bölümleri sayısına bölünmesiyle ölçülür ve genellikle yüzde olarak ifade edilir.
- Örnek PESEL doğrulama sistemini tekrar ele alalım.
- Bölümlemenin Şekildeki gibi göründüğünü ve aşağıdaki test durumlarını tanımladığımızı varsayalım:
  - 1898 doğumlu bir yetişkinin PESEL'i
  - 1999 doğumlu bir yetişkinin PESEL'i
  - PESEL = "1234" (çok kısa)
- Üç test senaryosundan oluşan bu set, belirlenen 12 eşdeğerlik bölümünden 3'ünü kapsar, böylece şu sonuca ulaşır: 3/12 = 1/4 = %25 kapsamı.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Birden Fazla Eşdeğerlik Bölümünün Aynı Anda Kapsanması: Kusur Maskeleme

- Testlerimizin aynı anda birden fazla alandan türetilen eşdeğerlik bölümlerini kapsayacağı bir durumda biraz farklı bir yaklaşıma ihtiyaç vardır.
- Böyle bir durumda iki veya daha fazla geçersiz bölümü kapsayacak test senaryoları oluşturmamak iyi bir uygulamadır.
- Bunun sözde kusur maskelemeyle ilgisi vardır.
- Önerilen strateji aşağıdaki gibidir:
  - İlk olarak, yalnızca geçerli bölümlerdeki test verilerinden oluşan ve tüm etki alanlarındaki tüm geçerli bölümleri kapsayacak mümkün olan en küçük sayıda test senaryosunu oluşturun.
  - Ardından, ortaya çıkarılan her geçersiz bölüm için, o bölümdeki verilerin yer alacağı ve diğer tüm verilerin geçerli bölümlerden geleceği ayrı bir test durumu oluşturun.
- Bu yaklaşımı açıklamak için aşağıdaki örneği inceleyin.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Birden Fazla Eşdeğerlik Bölümünün Aynı Anda Kapsanması: Kusur Maskeleme

- Örnek: Sistem öğrenciye iki veriye dayanarak not verir: alıştırmalar için puan sayısı (0-50) ve sınav için puan sayısı (0-50).
- Toplam puan 50'yi geçerse öğrenci dersi geçer.
- Dolayısıyla test senaryosunun girdi verileri iki bölümden oluşacaktır: alıştırma puanları ve sınav puanları.
- Aşağıdaki etki alanlarını ve bunların bölümlerini ayırt ettiğimizi varsayalım:

- A değişkeninin alanı = “Alıştırma puanları”:
  - (A1) geçersiz bölüm: 0'dan küçük sayılar
  - (A2) geçerli bölüm: 0'dan 50'ye kadar sayılar
  - (A3) geçersiz bölüm: 50'den büyük sayılar

- B değişkeninin alanı = “sınav puanları”:
  - (B1) geçersiz bölüm: 0'dan küçük sayılar
  - (B2) geçerli bölüm: 0'dan 50'ye kadar sayılar
  - (B3) geçersiz bölüm: 50'den büyük sayılar

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Birden Fazla Eşdeğerlik Bölümünün Aynı Anda Kapsanması: Kusur Maskeleme

- A'dan iki ve B'den iki geçersiz bölüm kaldı.
- Bu nedenle, bu bölümlerin ayrı ayrı test edileceği dört test senaryosuna daha ihtiyacımız var (belirli bir test senaryosundaki diğer bölüm, geçerli bölüm olmalıdır):
  - TC2: A = -8, B = 35 (geçersiz bölüm A1'i kapsar; ek olarak B2'yi kapsar)
  - TC3: A = 48, B = -11 (geçersiz bölüm B1'i kapsar; ek olarak A2'yi kapsar)
  - TC4: A = 64, B = 4 (geçersiz bölüm A3'ü kapsar; ayrıca B2'yi de kapsar)
  - TC5: A = 12, B = 154 (geçersiz bölüm B3'ü kapsar; ayrıca A2'yi kapsar).
- Her iki değerin de geçersiz bölümlerden geldiği bir durumu test ediyor olsaydık, kusur maskeleme olgusu ortaya çıkabilir. Sistemin bir öğrencinin bir dersi geçtiğini aşağıdaki prosedür yoluyla (sözde kodda açıklanmıştır) doğruladığını varsayalım:

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Birden Fazla Eşdeğerlik Bölümünün Aynı Anda Kapsanması: Kusur Maskeleme

- A'dan iki ve B'den iki geçersiz bölüm kaldı.
- Bu nedenle, bu bölümlerin ayrı ayrı test edileceği dört test senaryosuna daha ihtiyacımız var (belirli bir test senaryosundaki diğer bölüm, geçerli bölüm olmalıdır):
  - TC2: A = -8, B = 35 (geçersiz bölüm A1'i kapsar; ek olarak B2'yi kapsar)
  - TC3: A = 48, B = -11 (geçersiz bölüm B1'i kapsar; ek olarak A2'yi kapsar)
  - TC4: A = 64, B = 4 (geçersiz bölüm A3'ü kapsar; ayrıca B2'yi de kapsar)
  - TC5: A = 12, B = 154 (geçersiz bölüm B3'ü kapsar; ayrıca A2'yi kapsar).
- Her iki değerin de geçersiz bölümlerden geldiği bir durumu test ediyor olsaydık, kusur maskeleme olgusu ortaya çıkabilir. Sistemin bir öğrencinin bir dersi geçtiğini aşağıdaki prosedür yoluyla (sözde kodda açıklanmıştır) doğruladığını varsayalım:

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Birden Fazla Eşdeğerlik Bölümünün Aynı Anda Kapsanması: Kusur Maskeleme

- Şimdi aşağıdaki test senaryosunu göz önünde bulundurun:

```
INPUT: ExercisesPoints, ExamPoints
IF (ExercisesPoints + ExamPoints > 50) THEN
  RETURN "Course passed."
ELSE
  RETURN "Course failed."
```

- TC6: A = -28, B = 105 (geçersiz A1 ve B3 bölümlerini kapsar).

- Bu durumda toplam puan -28 + 105 = 77 olacak ve bu nedenle her iki giriş de yanlış olmasına rağmen sistem "Ders geçildi" sonucunu verecektir!

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

“Her Seçim” Kapsamı

- "Her seçim" kapsamı, çok boyutlu duruma, yani birden fazla alanın olduğu ve her test durumunun, her alanın dağılımından bir bölümü kapsadığı duruma uygulanır.
- Bu kapsam, çok boyutlu duruma uygulanan en basit (ve en zayıf) kapsam türlerinden biridir.
- Her alanın her bölümünün en az bir kez test edilmesini gerektirir.
- Uygulamada, bu yöntemi kullanarak, test uzmanı bir sonraki test senaryosunun mümkün olduğunca önceden ortaya çıkarılmamış kapsam öğelerini kapsamasını sağlamaya çalışır.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

“Her Seçim” Kapsamı

- Örnek: Bir COTS ürününü genel satış için test ediyoruz.
- Bu, onu farklı ortamlarda test etmemiz gerektiği anlamına gelir.
- Program farklı işletim sistemleri ve farklı tarayıcılarla çalışır.
- Bu nedenle farklı tarayıcıların farklı işletim sistemleri altında çalışmasını test etmek gerekir.
- Test etmek için aşağıdaki dört tarayıcıya ve üç işletim sistemine sahip olduğumuzu varsayalım:
  - Google Chrome (GC) • Firefox (F) • Safari (S) • Opera (O)
  - Windows (W) • Linux (L) • iOS

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

“Her Seçim” Kapsamı

- Basitlik açısından, işletim sistemlerinin veya tarayıcıların belirli sürümlerini eklemiyoruz.
- Her tarayıcı türü tek öğeli bir eşdeğerlik bölümü oluşturur ve sonuçta dört bölüm oluşur: {GC}, {F}, {S} ve {O}.
- Her işletim sistemi sırasıyla tek öğeli bir eşdeğerlik bölümü oluşturarak toplamda üç bölüm oluşturur: {W}, {L} ve (iOS}.
- “Her seçim” kapsam kriterine göre tanımlanan her bir bölüm için, o bölümdeki bir değeri kapsayan bir test senaryosu bulunmalıdır.
- Örneğimizde test durumları bir çift test verisi (tarayıcı türü, işletim sistemi) ile temsil edilir.
- Örneğimizde kapsam kriterini karşılamak için dört test senaryosu yeterlidir, örneğin:
- TC1: (GC, W) TC2: (F, L) TC3: (S, iOS) TC4: (O, W)

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Tespit Edilen Sorun Türleri

- EP tekniği, hatalı veri işlemeden, yani alan modelindeki hatalardan kaynaklanan sorunları tanımlar.

Etki Alanını Tanımlamak Önemlidir

- EP tekniği her zaman sorunu modelleyen belirli bir alana uygulanır. Bazen alan adının seçimi açıktır, ancak bazen konu biraz daha karmaşık olabilir. Aşağıdaki oldukça tipik örneği düşünün.

Örnek

- Termostat, sıcaklık (tam derece olarak hesaplanır) 21 dereceyi aştığında ısıtmayı kapatır, sıcaklık 18 derecenin altına düştüğünde açılır. Eşdeğerlik bölümlemeyi kullanarak test senaryoları tasarlayın.
- EP tekniği bu soruna nasıl uygulanır? Kontrol edilecek kaç tane denklik bölümü olacak? Bu, sistemin hangi spesifik özelliğini kontrol etmek istediğimize bağlıdır.

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Örnek

- Sorunumuza en az üç şekilde yaklaşabiliriz:
  - Yöntem 1. Yalnızca etki alanını analiz ettiğimizde, 22 ve üzeri değerlerde sistemin ısıtmayı kapattığını fark ederiz; 17 ve altındaki değerler için açılır; 18 ile 21 arasındaki değerler için ise herhangi bir işlem yapılmaz. Bu nedenle, "sıcaklık" alanı için 17 dereceye kadar, 18'den 21 dereceye kadar ve 21 derecenin üzerinde olmak üzere üç eşdeğerlik bölümümüz var ve bunları kapsamak için üç test senaryosuna ihtiyacımız var, örneğin:

  - Sıcaklık = 15 (beklenen çıktı: ısıtma açık).
  - Sıcaklık = 20 (beklenen çıktı: ısıtma durumu önceki duruma göre değişmez).
  - Sıcaklık = 22 (beklenen çıktı: ısıtma kapalı).

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Örnek

- Sorunumuza en az üç şekilde yaklaşabiliriz:
  - Yöntem 2. 18–21 sıcaklık aralığında ısıtmanın hem açık hem de kapalı olabileceğini unutmayın. Bu nedenle çiftlerden (sıcaklık, ısıtma durumu) oluşan bir alanı ele alıyoruz. Bu durumda dört ihtimalimiz var:
    - Sıcaklık <18, ısıtma açık
    - Sıcaklık 18–21, ısıtma açık
    - Sıcaklık 18–21, ısıtma kapalı
    - Sıcaklık >21, ısıtma kapalı
  - Yani ele almamız gereken dört durum var. Durum artık eskisinden biraz daha karmaşık çünkü testi yürütmeden önce 18-21 sıcaklıklar için uygun ısıtma koşulunu zorlamamız gerekiyor. Yukarıda bahsedilen dört bölümü kapsayan test senaryoları bu nedenle şöyle görünebilir:
    - TC1: sıcaklık = 15 (beklenen çıkış, ısıtma açık).
    - TC2: sıcaklık = 17'den yükseldikten sonra 18 (beklenen çıkış, ısıtma hala açık).
    - TC3: sıcaklık = 22'den düştükten sonra 21 (beklenen çıkış, ısıtma hala kapalı).
    - TC4: sıcaklık = 22 (beklenen çıkış, ısıtma kapalı).

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Örnek

- Sorunumuza en az üç şekilde yaklaşabiliriz:
  - Yöntem 3: Statik çiftleri (sıcaklık, ısıtma durumu) değil, bu çiftler arasındaki geçişleri dikkate alabiliriz. Daha sonra alanımız çiftler (sıcaklık, ısıtma durumu) arasındaki olası geçişleri tanımlayacak ve altı olası öğeden oluşacaktır (aşağıdaki alan öğeleri listesinde sayılar sıcaklıkları belirtir ve KAPALI ve AÇIK sırasıyla ısıtmanın kapalı ve ısıtmanın açık olduğunu belirtir):

    - (17, AÇIK) → (18, AÇIK)
    - (18, AÇIK) → (17, AÇIK)
    - (18, KAPALI) → (17, AÇIK)
    - (21, AÇIK) → (22, KAPALI)
    - (21, KAPALI) → (22, KAPALI)
    - (22, KAPALI) → (21, KAPALI)

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Örnek

- Sorunumuza en az üç şekilde yaklaşabiliriz:
  - Dolayısıyla bu geçişleri simüle etmek için altı test senaryosuna ihtiyacımız var; örneğin ilk eleman için sistemin başlangıç konfigürasyonu 17 derecedir ve ısıtma durumu açıktır.
  - Test, sıcaklığı 18 dereceye çıkarmak ve ısıtmanın kapatılıp kapatılmadığını görmektir.
    - (17, AÇIK) → (16, AÇIK)
    - (19, AÇIK) → (20, AÇIK)
    - (19, KAPALI) → (18, KAPALI)
    - (22, KAPALI) → (23, KAPALI)

4.2.1 Equivalence Partitioning (EP – Eş Değer Aralık Testi)

Örnek

- Yukarıdaki örnekte de görebileceğiniz gibi test hedefi, alanın biçimini önemli ölçüde etkilemektedir.
- Bizim durumumuzda alan üç farklı şekilde modellendi: sıcaklıkları temsil eden sayılar kümesi olarak, çiftler kümesi (sıcaklık, ısıtma durumu) ve bu çiftler arasındaki geçişler kümesi olarak.
- EP tekniğinin bunların her birine uygulanması, sistemin önemli ölçüde farklı bir yönünün doğrulanmasıyla sonuçlandı.
- Bu nedenle bu tekniği uygulamadan önce üzerinde çalışacağımız alanın formunun tam olarak ne olduğunu her zaman bilmemiz gerekir.
- Ayrıca bazen beklenen test sonucunu tam olarak belirtme ihtiyacının, spesifikasyondaki hataları veya belirsizlikleri tespit etmemize olanak sağladığını da unutmayın.
- Yukarıdaki yöntem 1'de, bölüm 18-21'den gelen veriler için beklenen sonucun ne olması gerektiği çok açık değildir: termostatın açık mı yoksa kapalı mı olması gerektiği.
- Bu, çözmeye çalıştığımız sorunun kötü bir şekilde ortaya konduğunu gösterebilir.
- Bu örneğin analizinin sonunda çok önemli bir şeye daha değinelim. Yani, bu özel problem için eşdeğerlik bölümleme tekniği iyi bir seçim değildir (uygularken karşılaştığımız problemler bunu çok iyi göstermektedir).
- Bunun nedeni, EP'nin (statik) alanın uygulanmasındaki sorunları tespit etmeye odaklanmasıdır ve burada ele aldığımız test problemindeki temel konu, termostatın davranışının doğrulanmasıdır.
- Durum geçiş testi daha uygun bir teknik gibi görünmektedir.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

EP Sınır değer analizinin bir uzantısı olarak BVA

- EP tekniği üzerine kurulmuş bir tekniktir.
- Dolayısıyla çok yönlülüğü ve tespit edilen sorunların türü ikincisine benzer olacaktır.
- EP'nin farkı, BVA'da test için EP'nin çok spesifik öğelerini, yani bu bölümlerin sınırlarında bulunanları seçmemizdir.

Sınır Değerler: BVA Hangi Alanlar İçin Uygulanabilir?

- Bir bölümün sınır değeri, o bölümün en küçük veya en büyük öğesidir.
- "En küçük" ve "en büyük" ten bahsetmek ancak alanın elemanları üzerinde bir sıra ilişkisi tanımlarsak anlamlı olur.
- Bu nedenle, BVA tekniği yalnızca öğeleri bazı sıra ilişkilerine (örneğin, "<" ilişkisine sahip sayı kümelerine) göre sıralanan alanlara uygulanabilir.
- Bu tür alanlara örnek olarak doğal, tam sayı veya gerçek sayı kümeleri, zamana veya tarihe ilişkin değerler vb. verilebilir.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

- Sınır değerleri her zaman belirli bir eşdeğerlik bölümü için tanımlanır.
- Örneğin 1 ile 120 arasındaki doğal sayıları içeren “yaş” bölgesi {1, 2, ..., 18} (çocuk) ve {19, 20, ..., 120} (yetişkin), bu durumda “çocuk” bölümünün sınır değerleri 1 (en küçük) ve 18'dir (en büyük). “Yetişkin” bölümü için sınır değerleri 19 ve 120'dir.
- Ek olarak, bir şeyi daha varsaymalıyız: dikkate alınan eşdeğerlik paylarında "boşluklar" olamaz, yani resmi olarak konuşursak, a < b aynı eşdeğerlik bölümüne ait olacak şekilde iki a ve b değeri varsa, o zaman tüm ara elemanlar c, yani a < c < b'nin de aynı bölüme ait olması gerekir.
- Neden? Daha önce bahsedilen örneği düşünün, ancak 4. öğeyi "çocuk" bölümünden çıkarın.
- Bu bölüm {1, 2, 3, 5, 6, ..., 18} biçimindedir.
- Uç değerleri elbette hala 1 ve 18'dir, ancak şu soru ortaya çıkıyor: 3 ve 5 değerleri de bu bölümün "sınırları" değil mi?
- Sonuçta her ikisi de bölüme ait olmayan bir öğeye bitişiktir!
- Bu durumda bölümümüzü {1, 2, 3} ve {5, 6, ..., 18} olarak ikiye bölmemiz gerekir ve 4'ün değeri ayrı bir bölüm olur.
- Yani alan adı dört bölüme bölünecektir: {1, 2, 3}, {4}, {5, 6, ..., 18} ve {19, ..., 120}.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

BVA ile Test Durumlarının Türetilmesi

- BVA tekniğini kullanarak test senaryolarını türetmeye yönelik genel prosedür aşağıdaki gibidir:
  - Analiz etmek istediğiniz alanı tanımlayın.
  - Bu etki alanının eşdeğerlik bölümlerine denklik bölümlemesini gerçekleştirin.
  - Tanımlanan her bir eşdeğerlik bölümü için sınır değerlerini belirleyin (not: bazen analiz yalnızca belirli bölümlerle sınırlandırılabilir ve belirlenen tüm eşdeğerlik bölümlerini dikkate almak zorunda kalmayabilir).
  - Her bir sınır değeri için, bu sınır değerine ilişkin kapsam öğelerini (test edilecek unsurlar) belirleyin.
- İlk iki adım basitçe EP tekniğini uygular. 3. adımda bölümlerin sınırlarını belirliyoruz. 4. adımda belirlenen sınır değerlerine göre test girdi verileri olarak test senaryolarında kullanılacak unsurları belirliyoruz.
- BVA tekniğinde sınır değerlerinin (test koşulları), test için seçilecek öğelerle (kapsam öğeleri) mutlaka aynı olması gerekmez.
- Bu, hangi bölümleri düşündüğümüze ve BVA yönteminin seçilen çeşidine bağlı olacaktır.
- Bunun nedeni BVA'nın iki ana çeşidinin bulunmasıdır. Bunlar;
  - 2 değerli BVA
  - 3 değerli BVA olarak adlandırılır.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

2-Değerli BVA

- 2 değerli BVA'da tanımlanan her sınır değeri için o değer ve sınır değerinin ait olduğu bölüme ait olmayan en yakın komşusu test için seçilir.

- Örneğin, tam sayı alanındaki P = {1, 2, 3, 4, 5, 6} bölümünün sınırlarını (yani 1 ve 6 değerlerini) dikkate alırsak, test için şunu seçeriz:

  - 1 değeri (P'nin sınır değeri olarak)
  - 0 değeri (P'ye ait olmayan 1'in en yakın komşusu olarak)
  - 6 değeri (P'nin sınır değeri olarak)
  - 7 değeri (P'ye ait olmayan 6'nın en yakın komşusu olarak)

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

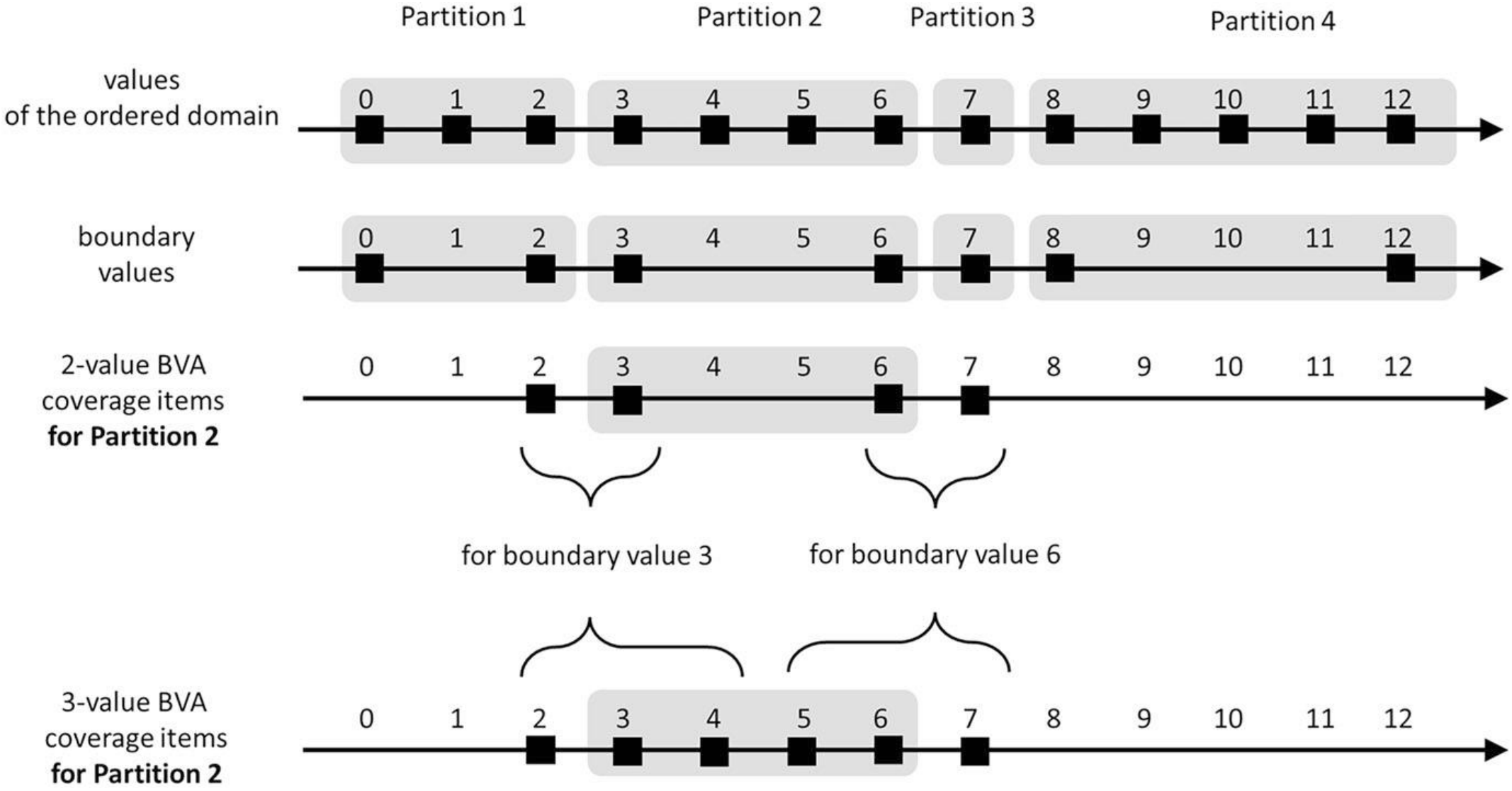
3-Değerli BVA

- Tanımlanan her sınır değeri için 3 değerli BVA'da, hangi bölüme ait olduklarına bakılmaksızın bu değeri ve onun her iki komşusunu test için alırız.

- Yukarıdaki örnekte test için şunu seçeceğiz:

  - 1 değeri (P'nin sınır değeri olarak)
  - 0 değeri (1'in sol komşusu olarak)
  - 2 değeri (1'in sağ komşusu olarak)
  - 6 değeri (P'nin sınır değeri olarak)
  - 5 değeri (6'nın sol komşusu olarak)
  - 7 değeri (6'nın sağ komşusu olarak)

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)



4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

Örnek

- Sistem, 18 yaş altı (çocuk indirimi) ve 65 yaş üstü (yaşlı indirimi) yolculara bilet indirimi uygulamaktadır.
- 18-65 yaş arası yolcu indirimden yararlanamaz.
- İndirim tahsisinin doğruluğunu kontrol etmek istiyoruz.
- Teminat kalemlerini belirlemek için açıklanan dört adımlı prosedürü uygulayalım.
  - Adım 1. Etki alanını tanımlayın. Analiz edilecek değişken, negatif olmayan bir tam sayı olan yolcunun yaşıdır. Bu nedenle alan adı {0, 1, 2, 3, ...} biçimindedir.
  - Adım 2. Eşdeğerlik bölümlerini tanımlayın. Spesifikasyondan aşağıdaki eşdeğerlik bölümlerini tanımlıyoruz:
    - P1: çocuk indirimi için uygun yaş {0, 1, 2, ..., 17}
    - P2: normal bilet için uygun yaş {18, 19, ..., 64, 65}
    - P3: yaşlılar için uygun yaş indirim {66, 67, 68, ...}
  - Adım 3. Sınır değerlerini tanımlayın:
    - P1'in sınır değerleri 0 ve 17'dir.
    - P2'nin sınır değerleri 18 ve 65'tir.
    - P3'ün sınır değeri 66'dır (bu örnekte P3 en büyük değere sahip değildir).

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

Örnek

- Adım 4. Test edilecek değerleri belirleyin.
  - Tüm eşdeğerlik paylaştırmaları için 2 değerli BVA'yı uygulamak istiyorsak aslında belirlenen tüm sınır değerlerinin test için alınması gerekir ve bu yeterlidir.
  - Bunun nedeni, iki bölümün herhangi iki komşu sınır değeri arasında bir tür simetriye sahip olmamızdır.
  - Örneğin, P2'nin sınır değeri olarak 65'in, bölümün dışından bir komşusu 66 vardır, bu da P3'ün sınır değeridir ve başka bir bölümden komşusu 65'tir.
  - Bu nedenle, test için 2 değerli BVA'yı kullanarak, tanımlanan tüm sınır değerlerini seçin: 0, 17, 18, 65 ve 66. Burada -1 değerinin uygun olmadığını varsayıyoruz.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

Örnek

- 3 değerli BVA durumunda 0, 1, 16, 17, 18, 19, 64, 65, 66 ve 67'yi test etmeyi seçiyoruz çünkü:
  - 0 sınır değeri için bu değeri ve komşusunu alıyoruz: 0 ve 1 (-1 değerinin uygun olmadığını varsayıyoruz).
  - 17 sınır değeri için bu değeri ve komşularını alıyoruz: 16, 17 ve 18.
  - 18 sınır değeri için bu değeri ve komşularını alıyoruz: 17, 18 ve 19.
  - 65 sınır değeri için bu değeri ve komşularını alıyoruz: 64, 65 ve 66.
  - 66 sınır değeri için bu değeri ve komşularını alıyoruz: 65, 66 ve 67.

- Elbette bazı değerler tekrarlanıyor ancak her değeri yalnızca bir kez test etmek için alıyoruz.
- Analizimize yalnızca orta bölüm ("normal bilet") konu olsaydı, yalnızca bu bölümün sınırlarını belirlemekle sınırlı olurduk, yani 18 ve 65 değerleri.
- Yani 2 değerli BVA'da test için 17, 18, 65 ve 66 değerlerini seçeceğiz ve 3 değerli BVA'da 17, 18, 19, 64, 65 ve 66 değerlerini seçeceğiz.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

Örnek

- Örnek Kodun bir yerinde x tamsayı değişkeninin değerine bağlı olarak bir karar verilir. Karar şu şekilde olmalıdır:

```
IF x < 16 THEN
   Perform ACTION 1
ELSE
   Perform ACTION 2
```

- Bu durumda, iki eşdeğerlik bölümü vardır: birincisi, "EYLEM 1"in yürütülmesine neden olan, 16'dan küçük tamsayılar olan değerleri içerir; ikincisi ise onun tamamlayıcısıdır yani “EYLEM 2”nin yürütülmesine neden olan 16'dan büyük veya ona eşit değerler.
- İlk bölümün sınırı 15 ve ikinci bölümün sınırı 16'dır. X değişkeni ne en küçük ne de en büyük değere sahiptir, dolayısıyla birinci bölümün minimum sınır değeri yoktur ve ikinci bölümün maksimum sınır değeri yoktur.
- Böylece, 2 değerli BVA'da test için 15 ve 16'yı seçeceğiz ve 3 değerli BVA'da 14, 15, 16 ve 17'yi seçeceğiz.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

Sınır Değerlerinin Dikkatle Belirlenmesi

- BVA durumunda, bölüm sınırlarının nasıl tanımlandığı konusunda çok dikkatli olunmalıdır.
- Diyelim ki doğal sayılar alanında çalışıyoruz.
- O halde, örneğin “7'den az olmayan öğeler içerir” formülasyonu, bu bölüme ait en küçük sayının 7 olduğu anlamına gelir.
- Buna karşılık, “7'den büyük öğeler içerir” formülasyonu, bu bölümden en küçük değerin olduğu anlamına gelir ve 8 değeri sınırıdır.
- Benzer şekilde “en fazla 65” 65’e kadar olan sayıları, “x < 65” koşulu ise bu eşitsizliği sağlayan en büyük değerin 64 olduğunu ifade eder.
- "x'ten y'ye kadar olan değerleri içerir" ifadesi, bölümün x ve y dahil x'den y'ye kadar olan değerleri içerdiği anlamına gelir.
- Bu konuda şüpheleri olanların şu soruyu dikkate almaları tavsiye ediliyor: Mağaza hangi günler açık ve üzerinde "Mağaza Pazartesi'den Cuma'ya kadar açık" yazan bir kağıt asılı. Mağaza pazartesiden cumaya haftanın 5 günü mü yoksa yalnızca salıdan perşembeye kadar mı açık?

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

BVA'nin Uygulanması

- BVA yöntemi çok basit ancak belirli kusur türlerinin tespitinde son derece etkilidir.
- Bunun nedeni, geliştiricilerin sıklıkla "birer birer hatalar" olarak adlandırılan, denklik bölümü sınırlarını yanlış uygulayarak yapmalarıdır.
- BVA tarafından tespit edilebilen kusurlarla sonuçlanan tipik geliştirici hatalarına iki örnek:
  - Geliştiricinin "if x < 10" (katı eşitsizlik) koşulunu uygulaması gerekirken, yanlışlıkla "if x ≤ 10" (zayıf eşitsizlik) şeklinde uygulaması.
  - Geliştiricinin döngü kontrol değişkenini (yineleyici) "for i=0 to 10" olarak başlatması gerekirken, yanlışlıkla dizinin elemanlarının 1'den indekslendiğini varsayarak bunu "for i=1 to 10" olarak yazması.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

BVA'nin Uygulanması

- İlk durumda, spesifikasyona göre eşdeğerlik bölümleri şu şekilde görünmelidir: {..., 8, 9}, {10, 11, ...}.
- Ancak hatalı uygulama, alanı (gerçekleştirilen kontrol akışı nedeniyle) şu şekilde böldü: {..., 9, 10}, {11, 12, ...}.
- Bu örneği BVA ile analiz ettiğimizde sınır değerlerinin (spesifikasyona göre geçerli) 9 ve 10 olduğunu görüyoruz.
- 2 değerli BVA kullanırsak test için bu değerleri alırız.
- Şartnameye göre değerin 10 olması durumunda programın "if x < 10" kararında "false" dalına karşılık gelen yolu izlemesi gerekir. X için test veri değeri olarak 10'u kullanan test senaryosunda bu kusurun tespit edilmesi ihtimali yüksektir.
- 3 değerli BVA, 2 değerli BVA'dan daha güçlüdür, yani 2 değerli BVA'nın bir arızayı tespit etme şansının olmadığı, 3 değerli BVA'nın ise bir arızayı tetikleyebildiği durumlar vardır. Aşağıdaki örneği düşünün.

4.2.2 Boundary Value Analysis (BVA -Sınır Değer Analizi)

Etki Alanı Dışındaki Değerler

- Son olarak bir sorunu daha ele alalım.
- Bilet indirimi atama örneğinde, 0 sınır değeri için komşusu -1'i test için almadığımızı, çünkü bunun mümkün olmadığını, yani kullanıcının yanlış giremeyeceğini varsaydığımızı belirttik.
- Ancak pratikte bu her zaman böyle olmak zorunda değildir.
- Örneğin kullanıcı klavyeden bir form alanına yaş değeri girerse elbette negatif bir değer de girebilir.
- Aşırı, alan dışı sınır değerlerinin test edilip edilmeyeceği, arayüzün buna izin verip vermediğine bağlıdır.
- Eğer öyleyse, testçinin elbette bunları test etmesi gerekir.

4.2.3 Decision Table Testing(Karar Tablosu Testi)

Kapsam

- Karar tablosu testi, iş kuralları uygulamalarının doğruluğunu onaylamak için kullanılan bir tekniktir. Bir iş kuralı genellikle mantıksal bir çıkarım biçimini alır:
  - IF (condition) THEN (action),

- Hem koşulun hem de eylemin birden fazla faktörden oluşabileceği yerdir.
-
- Daha sonra koşulların veya eylemlerin bir kombinasyonuyla uğraşıyoruz. İşte iş kurallarına ilişkin bazı örnekler:

  - IF (customerAge < 18) THEN (assign a ticket discount);
  - IF (a+b>c>0 AND a+c>b>0 AND b+c>a>0) THEN (you can build a triangle with sides of length a, b, c);
  - IF (monthlySalary > 10000) THEN (grant bank loan AND offer a gold card).

4.2.3 Decision Table Testing(Karar Tablosu Testi)

Kapsam

- Karar tabloları, koşul kombinasyonlarının uygulanmasının doğruluğunu sistematik olarak test etmemize olanak tanır.

	Business rules 1–8							
	1	2	3	4	5	6	7	8
Conditions								
Has a loyalty card?	YES	YES	YES	YES	NO	NO	NO	NO
Total amount > $1000?	YES	YES	NO	NO	YES	YES	NO	NO
Shopping in last 30 days?	YES	NO	YES	NO	YES	NO	YES	NO
Actions								
Granted discount	10%	5%	5%	0%	0%	0%	0%	0%

4.2.3 Decision Table Testing(Karar Tablosu Testi)

Karar Tablosundan Test Durumlarının Çıkarılması

- Karar tablolarını kullanarak spesifik test senaryoları oluşturma süreci aşağıdaki beş adımda sunulabilir.

- Adım 1. Tüm olası tekil koşulları belirleyin ve bunları tablonun üst kısmındaki ardışık satırlarda listeleyin. Gerekirse bileşik koşulları tek koşullara bölün. Koşullar genellikle spesifikasyonlarda «if» vb. sözcüklerin önüne gelen cümle parçaları olarak görünür.

- Adım 2. Sistemde meydana gelebilecek ve bu koşullara bağlı olan (aynı zamanda test temelinden türetilen) tüm ilgili eylemleri tanımlayın ve bunları tablonun alt kısmındaki ardışık satırlarda listeleyin. Eylemler genellikle spesifikasyonlarda "o zaman", "bu durumda", "sistem şunu yapmalıdır" vb. kelimelerin ardından gelen cümle parçaları olarak görünür.

4.2.3 Decision Table Testing(Karar Tablosu Testi)

Karar Tablosundan Test Durumlarının Çıkarılması

- Adım 3. Tüm koşul kombinasyonlarını oluşturun ve uygun olmayan koşul kombinasyonlarını eleyin. Her uygun kombinasyon için, tabloda her bir koşulun değerlerinin bu sütunda listelendiği ayrı bir sütun oluşturulur.

- Adım 4. Belirlenen her bir koşul kombinasyonu için hangi eylemlerin ve bunların nasıl gerçekleşmesi gerektiğini belirleyin. Bu, karar tablosunun ilgili sütununun alt kısmının tamamlanmasıyla sonuçlanır.

- Adım 5. Karar tablosunun her sütunu için, test girdisinin bu sütunda belirtilen koşulların kombinasyonunu temsil ettiği bir test senaryosu tasarlayın. Test, yürütüldükten sonra sistem, tablonun alt kısmında ilgili sütunda açıklanan eylemleri gerçekleştirirse başarılı olur. Bu eylem girişleri, test senaryosu için beklenen çıktı görevi görür.

4.2.3 Decision Table Testing(Karar Tablosu Testi)

Karar Tablosunda Gösterim ve Olası Girişler

- Genellikle koşul ve eylem değerleri, True veya False mantıksal değerleri biçimini alır.
- Bunlar, T ve F veya Y ve N (evet/hayır) veya 1 ve 0 sembolleri gibi çeşitli şekillerde temsil edilebilirler.
- Ancak koşulların ve eylemlerin değerleri genel olarak sayılar, sayı aralıkları, kategori değerleri, eşdeğerlik bölümleri vb. gibi herhangi bir nesne olabilir.
- Örneğin, tablomuzda, "Granted discount" eyleminin değerleri, farklı indirim türlerini ifade eden kategorilerdir: %0, %5 ve %10.
- Aynı tabloda farklı türde koşullar ve eylemler olabilir; örneğin mantıksal, sayısal ve kategorik koşullar aynı anda ortaya çıkabilir.
- Yalnızca Boolean (doğru/yanlış) değerlerine sahip karar tablosuna sınırlı girişli karar tablosu adı verilir.
- Herhangi bir koşul veya eylem için Boole girişlerinden başka girişler varsa, böyle bir tabloya genişletilmiş girişli karar tablosu adı verilir.

4.2.3 Decision Table Testing(Karar Tablosu Testi)

Tüm Koşul Kombinasyonları Nasıl Belirlenir?

- Koşul kombinasyonlarını manuel olarak belirlememiz gerekiyorsa ve bazı kombinasyonları kaçıracağımızdan korkuyorsak, tüm koşul kombinasyonlarını sistematik olarak belirlemek için çok basit bir ağaç yöntemini kullanabiliriz. Aşağıdaki örneği düşünün:

- Örnek Bir karar tablosunun üç koşulu olduğunu varsayalım:

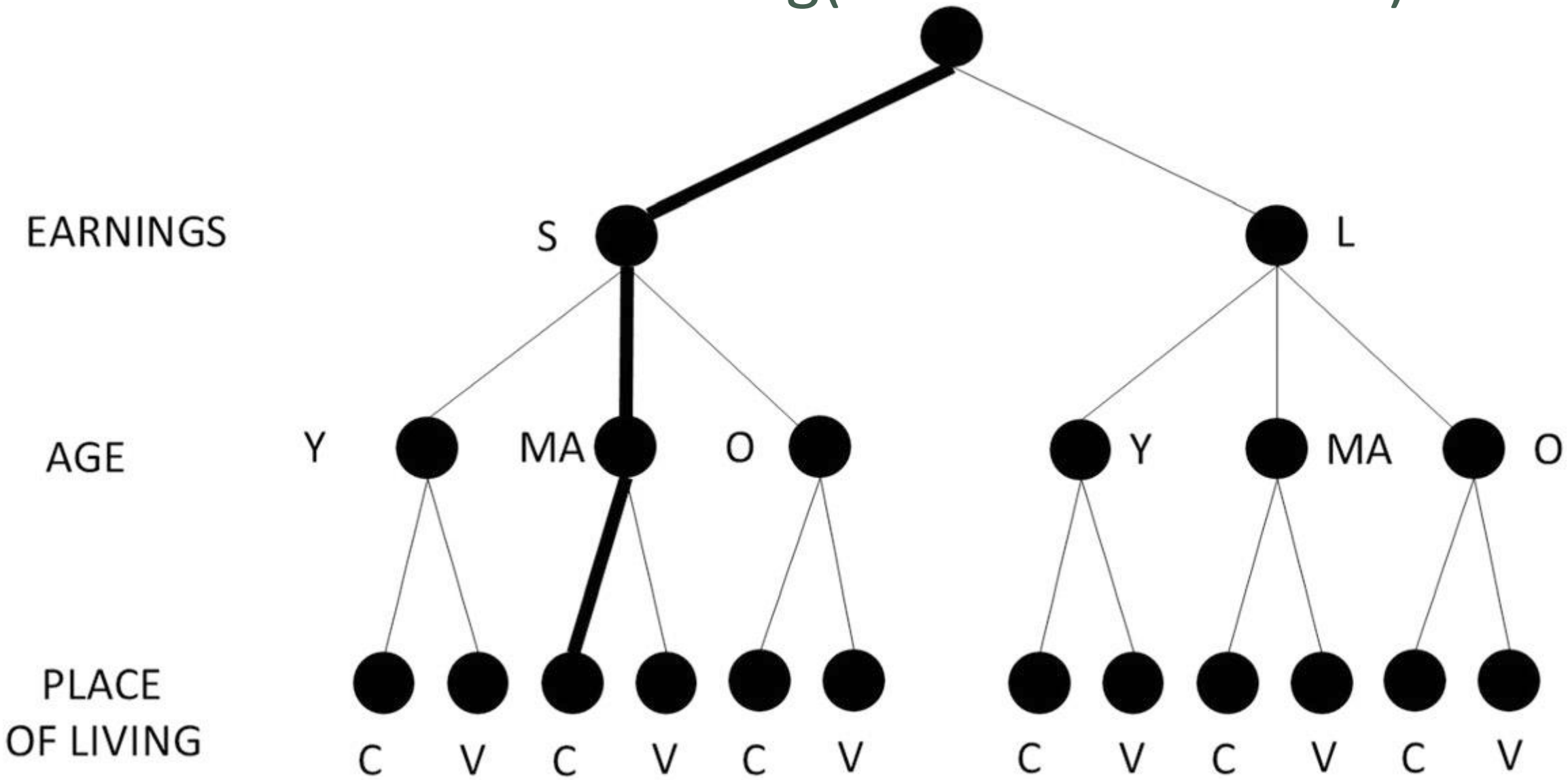
  - Earnings (two possible values—S, small; L, large)
  - Age (three possible values—Y, young; MA, middle aged; O, old)
  - Place of living (two possible values—C, city; V, village)

4.2.3 Decision Table Testing(Karar Tablosu Testi)

Tüm Koşul Kombinasyonları Nasıl Belirlenir?

- Üçlü değerlerin (age, earnings, residence) tüm kombinasyonlarını oluşturmak için, ilk koşulun (earnings) tüm olasılıklarını kökünden türettiğimiz bir ağaç inşa ediyoruz.
- Bu ağacın ilk seviyesidir.
- Daha sonra, bu seviyenin her köşesinden ikinci koşulun (age) tüm olasılıklarını türetiyoruz.
- Ağacın ikinci seviyesini alıyoruz.
- Son olarak, bu düzeyin her bir köşesinden üçüncü koşulun (place of living) tüm olası değerlerini elde ederiz.
- Elbette daha fazla şart olsaydı benzer şekilde hareket ederdik.

4.2.3 Decision Table Testing(Karar Tablosu Testi)



4.2.3 Decision Table Testing(Karar Tablosu Testi)

Statik Test Tekniği Olarak Karar Tabloları

- Karar tablosu testi, gereksinimlerin yokluğu veya çelişkisi gibi sorunları tespit etmek için mükemmeldir.

- Karar tablosu spesifikasyondan oluşturulduktan sonra veya hala oluşturuluyorken, aşağıdaki gibi spesifikasyon problemlerini keşfetmek çok kolaydır:

  - Eksiklik—belirli bir koşul kombinasyonu için tanımlanmış eylem yok
  - Çelişki—aynı koşullar kombinasyonuna karşı sistemin iki farklı davranışını iki farklı spesifikasyon yerinde tanımlamak
  - Artıklık— aynı sistem davranışını spesifikasyonda iki farklı yerde tanımlamak (belki farklı şekilde tanımlanır)

4.2.4 State Transition Testing - Durum Geçiş Testi

Uygulama

- Durum geçiş testi, bir bileşenin veya sistemin davranışını kontrol etmek için kullanılan bir tekniktir. Bu nedenle, davranışsal yönünü, yani zaman içinde nasıl davrandığını ve çeşitli olay türlerinin etkisi altında durumunu nasıl değiştirdiğini kontrol eder.

- Bu davranışsal yönü tanımlayan model, durum geçiş diyagramı olarak adlandırılır. Literatürde, bu modelin farklı varyantlarına sonlu otomasyon, sonlu durum otomasyonu, durum makinesi veya Etiketli Geçiş Sistemi adı verilir.

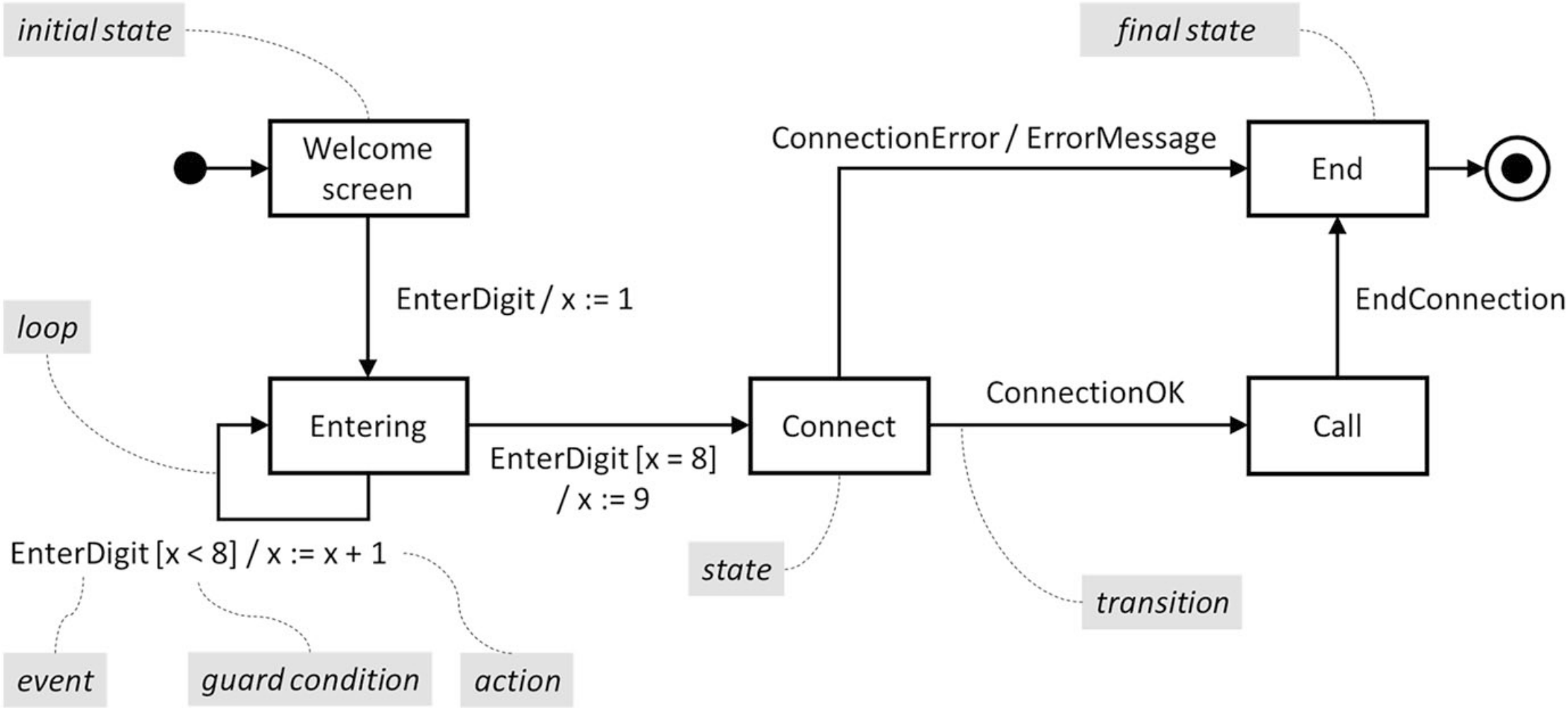
- Ders programı, durum geçiş modelinin grafiksel biçimini belirtmek için "durum geçiş diyagramı" adını ve modelin eşdeğer, tablo biçimindeki biçimini belirtmek için "durum geçiş tablosu" adını kullanır.

4.2.4 State Transition Testing - Durum Geçiş Testi

Durum Geçiş Diyagramının Oluşturulması

- Durum geçiş diyagramı, UML standardında açıklandığı gibi bir grafik modeldir. Teorik açıdan bakıldığında, yönlendirilmiş grafik olarak adlandırılır. Durum geçiş diyagramı aşağıdaki unsurlardan oluşur:
  - Durumlar—sistemin olabileceği olası durumları temsil eder
  - Geçişler—durumların olası (doğru) değişikliklerini temsil eder
  - Olaylar—genellikle sistemin dışında olan ve oluşumu ilgili geçişleri tetikleyen fenomenleri temsil eder
  - Eylemler—durumlar arasındaki geçiş sırasında sistemin alabileceği eylemler
  - Koruma koşulları—geçişlerle ilişkili mantıksal koşullar; bir geçiş yalnızca ilişkili koruma koşulu varsa yürütülebilir doğru
- Aşağıdaki diyagram, belirli bir numaraya sahip bir cep telefonu kullanıcısına telefon araması yapmak için sistemin davranışının bir modelini temsil eder. Kullanıcı, numaranın ardışık rakamlarına karşılık gelen tuşlara tek tek basarak tek haneli telefon numarasını çevirir. Dokuzuncu rakam girildiğinde sistem otomatik olarak aramayı dener.

4.2.4 State Transition Testing - Durum Geçiş Testi



4.2.4 State Transition Testing - Durum Geçiş Testi

Step	State	Event	Action	Next state
1	Welcome screen	EnterDigit	x := 1	Entering
2	Entering	EnterDigit	x := x + 1	Entering
3	Entering	EnterDigit	x := x + 1	Entering
4	Entering	EnterDigit	x := x + 1	Entering
5	Entering	EnterDigit	x := x + 1	Entering
6	Entering	EnterDigit	x := x + 1	Entering
7	Entering	EnterDigit	x := x + 1	Entering
8	Entering	EnterDigit	x := x + 1	Entering
9	Entering	EnterDigit	x := 9	Connect
10	Connect	ConnectionOK		Call
11	Call	EndConnection		End

Şekildeki durum geçiş diyagramındaki geçiş sırası örneği.

4.2.5 Use Case Testing - Kullanım Senaryosu Testi (Yeni Müfredatta Yok)

- Kullanım senaryosu, çoğunlukla kullanıcı ve sistem olan sözde aktörler arasındaki etkileşimi açıklayan bir gereksinim belgesidir.
- Kullanım durumu, kullanıcının ve sistemin gerçekleştirdiği ve sonuçta kullanıcının bir miktar fayda elde etmesine yol açan bir dizi adımın açıklamasıdır.
- Her kullanım senaryosu tek ve iyi tanımlanmış bir senaryoyu tanımlamalıdır.
- Örneğin, bir ATM'nin (para çekme makinesi) gereksinimlerini belgeliyorsak, kullanım senaryoları diğer şeylerin yanı sıra aşağıdaki senaryoları da açıklayabilir:
  - Geçersiz bir ATM kartının reddedilmesi
  - Doğru PIN'i girerek sisteme giriş yapılması
  - ATM'den doğru şekilde para çekilmesi
  - Hesapta yetersiz bakiye nedeniyle para çekme girişimi başarısız oldu
  - Üç kez yanlış PIN girilerek kartın kilitlenmesi
- Her kullanım durumu için test uzmanı, senaryo çalıştırması sırasında beklenmeyen olayların meydana gelip gelmediğini kontrol etmek için karşılık gelen bir test senaryosunun yanı sıra bir dizi test senaryosu da oluşturabilir.

4.2.5 Use Case Testing - Kullanım Senaryosu Testi (Yeni Müfredatta Yok)

Kullanım Senaryosu Yapısı

- Doğru şekilde oluşturulmuş bir kullanım örneği, benzersiz bir numara ve ad gibi "teknik" bilgilerin yanı sıra aşağıdakilerden oluşmalıdır:
  - Ön koşullar (giriş verileri dahil)
  - Tam olarak bir, sıralı ana senaryo
  - (İsteğe bağlı)Alternatif akışlar ve istisnalar olarak adlandırılan konumların işaretlenmesi
  - Son koşullar (senaryonun başarılı bir şekilde yürütülmesinden sonra sistemin durumunu ve elde edilen kullanıcı faydasını açıklar)

4.3 Beyaz Kutu Test Teknikleri

FL-4.3.1(K2)İfade testini açıklayın.

FL-4.3.2(K2)Dallanma testini açıklayın.

FL-4.3.3(K2)Beyaz kutu testinin değerini açıklayın.

4.3 Beyaz Kutu Test Teknikleri

- Beyaz kutu testi, test nesnesinin iç yapısına dayanır.
- Çoğu zaman, beyaz kutu testi, test nesnesinin modelinin kodun iç yapısı olduğu test tekniğidir.
- Bununla birlikte, beyaz kutu test tekniklerinin tüm test seviyelerine uygulanabileceğini unutmamak önemlidir, örneğin:
  - Bileşen testinde
  - Entegrasyon testinde
  - Sistem testinde
  - Kabul testinde
- Temel Düzey müfredatı iki beyaz kutu test tekniğini tartışmaktadır: ifade testi ve dal testi. Bu tekniklerin her ikisi de doğası gereği kodla ilişkilidir, dolayısıyla uygulama alanları temel olarak bileşen testidir.

4.3 Beyaz Kutu Test Teknikleri

- Bunlara ek olarak, aşağıdakiler gibi başka (ve genellikle daha güçlü) beyaz kutu teknikleri de vardır:
  - MC/DC testing
  - Multiple conditions testing
  - Loop testing
  - Basis path testing

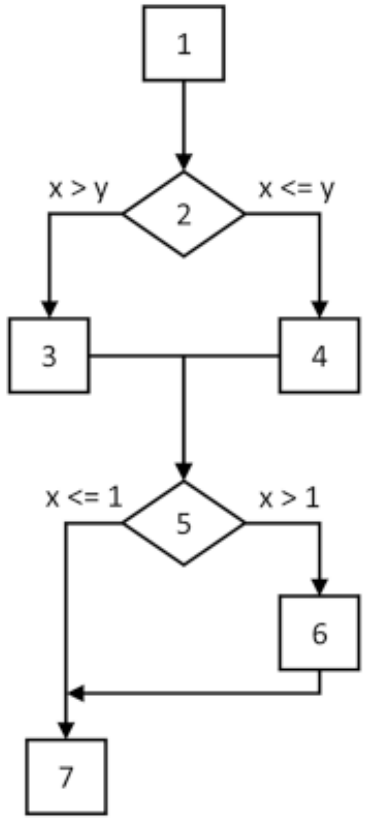
- Ancak bu teknikler Temel Düzey müfredatında yer almamaktadır.
- Bunlardan bazıları İleri Düzey Teknik Test Analisti müfredatında mevcuttur.
- Bu daha güçlü beyaz kutu test teknikleri genellikle güvenlik açısından kritik sistemleri test ederken kullanılır (örneğin, tıbbi cihaz yazılımı veya havacılık yazılımı).

4.3.1 İfade Testi ve İfade Kapsamı

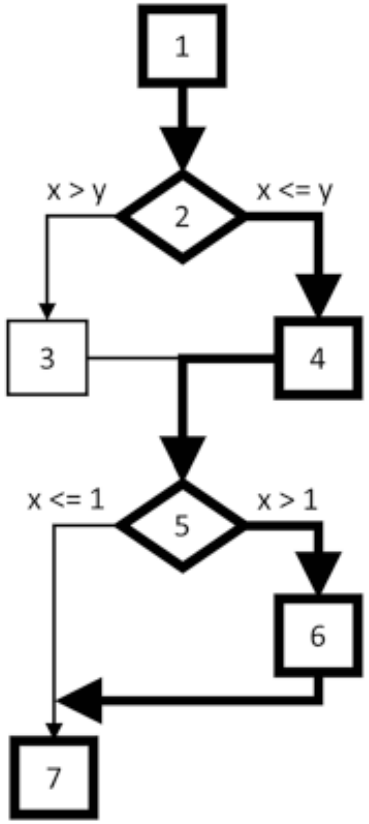
- İfade testi basit ve aynı zamanda en zayıf beyaz kutu test tekniğidir.
- Bir programın kaynak kodundaki yürütülebilir ifadeleri kapsamayı içerir.

```
1. INPUT x, y // two natural numbers
2.   IF (x > y) THEN
3.     z := x - y
     ELSE
4.     z := y - x
5.   IF (x > 1) THEN
6.     z := z * 2
7.   RETURN z
```

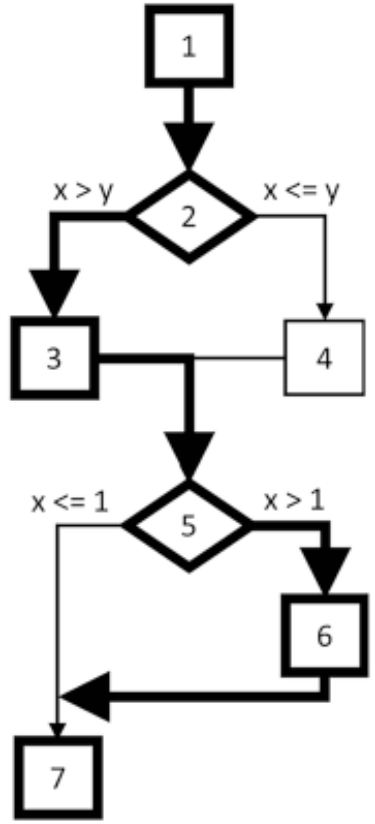
- TC1: giriş:x = 2, y = 11;beklenen çıkış:18.
- TC2: giriş:x = 8, y = 1;beklenen çıkış:14.



a) control flow graph



b) control flow for x = 2, y = 11



c) control flow for x = 8, y = 1

4.3.1 İfade Testi ve İfade Kapsamı

- Beyaz kutu test teknikleri bağlamında önemli olan her şeyi vurgulayalım: test senaryolarında beklenen sonuçlar koddan türetilmemiştir.
- Bunun nedeni, kodun tam olarak test ettiğimiz şey olmasıdır; yani kendi verisi olamaz.
- Beklenen sonuçlar kodun dışındaki bir spesifikasyondan elde edilir.
- Bizim durumumuzda şöyle olabilir: “Program iki doğal sayı alıyor ve sonra büyük olandan küçük olanı çıkarıyor.
- İlk sayı 1'den büyükse değer iki katına çıkar.
- Program bu şekilde hesaplanan değeri döndürür.”

4.3.2 Dal Testi ve Dal Kapsamı

- Dal, bir CFG'nin iki köşesi arasındaki kontrol akışıdır.
- Bir dal koşulsuz veya koşullu olabilir.
- A ve B köşeleri arasında koşulsuz bir dallanma, A ifadesinin yürütülmesi tamamlandıktan sonra kontrolün B ifadesine geçmesi gerektiği anlamına gelir.
- A ve B arasındaki koşullu dallanma, A ifadesinin yürütülmesi tamamlandıktan sonra kontrolün B ifadesine geçebileceği anlamına gelir, ancak bu zorunlu değildir.
- Koşullu dallar, karar köşelerinden, yani koddaki daha ileri kontrol seyrinin bağlı olduğu bazı kararların verildiği yerlerden çıkar.
- Karar bildirimine örnek olarak IF, IF-ELSE ve SWITCH-CASE koşullu ifadelerinin yanı sıra WHILE, DO-WHILE veya FOR döngülerindeki sözde döngü koşulunu kontrol eden ifadeler verilebilir.

4.4 Deneyime Dayalı Test Teknikleri

FL-4.4.1 (K2) Hata tahminini açıklayın.

FL-4.4.2 (K2) Keşif testini açıklayın.

FL-4.4.3 (K2) Kontrol listesine dayalı testleri açıklayın.

4.4 Deneyime Dayalı Test Teknikleri

- Kara ve beyaz kutu test tekniklerine ek olarak üçüncü bir teknik ailesi daha vardır: deneyime dayalı test teknikleri. Bu kategori, daha önce tartışılanlardan daha az resmi olduğu düşünülen teknikleri içerir.
- Deneyime dayalı test teknikleri öncelikle test uzmanlarının bilgi, beceri, sezgilerinden ve benzer ürünlerle, hatta aynı ürünle daha önce çalışma deneyimlerinden yararlanır.
- Bu yaklaşım, test uzmanının daha yapılandırılmış teknikler kullanılarak tespit edilmesi zor olan arızaları veya kusurları tespit etmesini kolaylaştırır.
- Görünüşe rağmen deneyime dayalı test teknikleri test uzmanları tarafından yaygın olarak kullanılmaktadır.
- Ancak, bu tekniklerin etkinliğinin doğası gereği büyük ölçüde bireysel test uzmanlarının yaklaşımına ve deneyimine bağlı olacağını akılda tutmak önemlidir.
- Temel Düzey ders programı, aşağıdaki bölümlerde tartışacağımız üç tür deneyime dayalı test tekniğini listelemektedir:
  - Hata tahmini
  - Keşif amaçlı testler
  - Kontrol listesine dayalı testler

4.4.1 Hata Tahmini

- Hata tahmini belki de kavramsal olarak tüm deneyime dayalı test tekniklerinin en basitidir.
- Herhangi bir ön planlamayla ilişkili değildir ve test uzmanının bu tekniğin kullanımına ilişkin faaliyetlerini yönetmek de gerekli değildir.
- Testi yapan kişi, diğer şeylerin yanı sıra aşağıdaki hususlara atıfta bulunarak, bir bileşen veya sistemdeki geliştiriciler tarafından yapılması muhtemel hata türlerini, kusurları veya arızaları basitçe tahmin eder:
  - Test edilen bileşen veya sistem (veya üretimde çalışmakta olan önceki sürümü) şu ana kadar nasıl çalıştı?
  - Testi yapan kişinin bildiği, geliştiriciler, mimarlar, analistler ve üretim ekibinin diğer üyeleri tarafından yapılan tipik hatalar nelerdir?

4.4.1 Hata Tahmini

- Test uzmanı tarafından test edilen benzer uygulamalarda daha önce meydana gelen veya test uzmanının genel olarak hatalar, kusurlar ve arızalar bulduğu noktalar aşağıdakilerle ilgili olabilir:
  - Giriş (örn. geçerli giriş kabul edilmedi, geçersiz giriş kabul edildi, yanlış parametre değeri, eksik giriş parametresi)
  - Çıkış (örn. yanlış çıkış formatı, hatalı çıkış, yanlış zamanda doğru çıkış, eksik çıkış, eksik çıkış, dilbilgisi veya noktalama işareti hataları)
  - Mantık (örneğin, dikkate alınacak eksik durumlar, dikkate alınacak yinelenen durumlar, geçersiz mantıksal operatör, eksik koşul, geçersiz döngü yinelemesi)
  - Hesaplamalar (örneğin, yanlış algoritma, etkisiz algoritma, eksik hesaplama, geçersiz işlenen, geçersiz operatör, basamaklama hatası , sonucun yetersiz hassasiyeti, geçersiz yerleşik fonksiyon)
  - Arayüz (örn. arayüzden olayların yanlış işlenmesi, giriş/çıkış işlemede zamana bağlı hatalar, yanlış veya var olmayan fonksiyonun çağrılması, eksik parametre, uyumsuz parametre türleri)
  - Veri (örneğin, bir değişkenin yanlış başlatılması, tanımlanması veya bildirilmesi, bir değişkene yanlış erişim, bir değişkenin yanlış değeri, geçersiz bir değişkenin kullanılması, bir veriye yanlış referans, yanlış ölçeklendirme veya veri birimi, Verinin yanlış boyutu, yanlış indeks, değişken tipinin yanlış olması, değişken aralığının yanlış olması)

4.4.2 Keşif Testi

- Keşifsel test, önceden yazılmamış testleri (yani önceden tasarlanmamış testleri) tasarlama, yürütme ve kaydetme ve bunları yürütme sırasında dinamik olarak değerlendirme yaklaşımıdır.
- Bu, test uzmanının ayrı bir tasarım veya analiz aşamasında hazırlanan testleri yürütmediği, ancak mevcut test senaryosundaki sonraki adımların her birinin dinamik olduğu ve aşağıdakilere bağlı olduğu anlamına gelir:
  - Testi yapanın bilgisi ve sezgisi
  - Bu veya benzer uygulamalarla ilgili deneyim
  - Senaryonun önceki adımında sistemin davranış şekli
- Aslında, keşifsel ve senaryolu testler arasındaki katı ayrım gerçeği tam olarak yansıtmıyor. Bunun nedeni, her test faaliyetinin bir miktar planlama unsuru ve bir miktar dinamik ve keşif unsuru içermesidir.
- Örneğin, saf keşif testinde bile, bir keşif oturumunu önceden planlamak, örneğin test uzmanına bunu yürütmesi için belirli bir zaman ayırmak olağandır. Keşif niteliğindeki bir test uzmanı, oturum başlamadan önce test için gereken çeşitli test verilerini de hazırlayabilir.

4.4.2 Keşif Testi

Oturum Tabanlı Keşif Testi

- Oturum bazlı keşif testi, test uzmanlarına ve test yöneticilerine, test uzmanının etkinliğinin daha iyi yapılandırılmasını sağlamanın yanı sıra, keşif amaçlı test faaliyetlerini daha iyi yönetme olanağı sağlayan bir yaklaşımdır.
- Üç temel adımdan oluşur:

1. Testi yapan kişi, testin kapsamını belirlemek ve süreyi belirlemek üzere yönetici ile görüşür. Yönetici, test başlatma belgesini test uzmanına verir veya bunu test uzmanıyla birlikte yazar. Test başlatma belgesi, test analizi sırasında oluşturulmalıdır.

2. Test uzmanı tarafından, test uzmanının ilgili gözlemleri (gözlenen sorunlar, geleceğe yönelik tavsiyeler, oturum sırasında neyin başarısız olduğuna ilişkin bilgiler, vb.) not ettiği bir keşif testi oturumunun yürütülmesi.

3. Oturumun sonunda, oturum sonuçlarının tartışıldığı ve sonraki olası adımlara ilişkin kararların alındığı yeniden toplantı.

4.4.2 Keşif Testi

Test charter—TE 02-001-01	
Goal	Test the login functionality
Areas	Log in as an existing user with a correct login and password Log in as an existing user through a Google account Log in as an existing user through a Facebook account Incorrect login—no user Incorrect login—wrong password Actions that result with the blocking of the account Using the password reminder function SQL injection attack[a] and other security attacks
Environment	Site accessible through various browsers (Chrome, FF, IE)
Time	2019-06-12, 11:00–13:15
Tester	Bob Bugbuster
Tester's notes	
Files	(Screenshots, test data, etc.).
Defects found	
Division of time	20% preparation for the session 70% conduct of session 10% problem analysis

4.4.3 Kontrol Listesi Tabanlı Test

- Kontrol listesi tabanlı test - bu bölümde açıklanan önceki iki teknik gibi - test uzmanının bilgi ve deneyimini kullanır, ancak testin yürütülmesinin temeli, kontrol listesinde yer alan öğelerdir.
- Kontrol listesi, doğrulanması gereken test koşullarını içerir.
- Kontrol listesi, otomatik olarak kontrol edilebilecek öğeler, giriş/çıkış kriterleri açısından daha iyi işlev gören öğeler veya çok genel öğeler içermemelidir.
- Kontrol listesi tabanlı testler, hatalı saldırılara benzer bir teknik gibi görünebilir.
- Bu teknikler arasındaki fark, hatalı saldırıların kusurlardan ve başarısızlıklardan başlaması ve test cihazının eyleminin, belirli bir arıza veya kusur göz önüne alındığında sorunları ortaya çıkarmak olmasıdır.
- Kontrol listesi tabanlı testte, test cihazı aynı zamanda sistematik bir şekilde hareket eder ancak yazılımın “olumlu” özelliklerini kontrol eder.
- Bu tekniğin test temeli, kontrol listesinin kendisidir.
- Kontrol listesi öğeleri genellikle soru biçiminde formüle edilir.
- Kontrol listesi, her bir tesis öğesini ayrı ayrı ve doğrudan kontrol etmenize izin vermelidir.
- Kontrol listesindeki öğeler gereksinimlere, kalite özelliklerine veya diğer en yumuşak koşullara atıfta bulunabilir.

4.4.3 Kontrol Listesi Tabanlı Test

- Kontrol listeleri, işlevsel ve işlevsel olmayan testler de dahil olmak üzere çeşitli test türlerini desteklemek için oluşturulabilir.
- Kontrol listesini geliştirenler aynı hataları yapmaktan kaçınmayı öğrendikçe, bazı kontrol listesi öğeleri zaman içinde giderek daha az etkili hale gelebilir.
- Yakın zamanda keşfedilen yüksek önemdeki kusurları yansıtmak için yeni öğelerin de eklenmesi gerekebilir.
- Bu nedenle kontrol listelerinin kusur analizlerine göre düzenli olarak güncellenmesi gerekmektedir.
- Ancak kontrol listesinin çok uzun olmamasına dikkat edilmelidir.
- Ayrıntılı test senaryolarının yokluğunda, kontrol listelerine dayalı testler, test için bir dereceye kadar tutarlılık sağlayabilir.
- Aynı kontrol listesiyle çalışan iki test uzmanı muhtemelen görevlerini biraz farklı şekilde gerçekleştireceklerdir, ancak genel olarak aynı şeyleri test edeceklerdir (aynı test koşulları), dolayısıyla testleri mutlaka benzer olacaktır.
- Kontrol listeleri yüksek düzeydeyse, gerçek testlerde muhtemelen bazı değişkenlikler olabilir ve bu da potansiyel olarak Daha yüksek kapsam, ancak testlerin daha az tekrarlanabilirliği.

4.5 İşbirliğine Dayalı Test Yaklaşımları

FL-4.5.1 (K2) Geliştiriciler ve iş temsilcileriyle işbirliği içinde kullanıcı hikayelerinin nasıl yazılacağını açıklayın.
FL-4.5.2 (K2) Kabul kriterlerinin yazılması için farklı seçenekleri sınıflandırın.
FL-4.5.3 (K2) Test senaryolarını türetmek için kabul testine dayalı geliştirmeyi (ATDD) kullanın.

4.5 İşbirliğine Dayalı Test Yaklaşımları

- Çevik metodolojilerin popülaritesi, bu metodolojiler tarafından kullanılan eserler dikkate alınarak ve müşteriler (işletme), geliştiriciler ve test uzmanları arasındaki işbirliğine vurgu yapılarak bu yaklaşıma özel test yöntemlerinin geliştirilmesine yol açmıştır.
- Bu bölümde, işbirliğine dayalı bir test yaklaşımı kullanılarak çevik metodolojiler bağlamında testlerle ilgili aşağıdaki konular tartışılmaktadır:
  - Kullanıcı gereksinimlerinin çevik bir karşılığı olarak kullanıcı hikayeleri
  - Test tasarımının temelini oluşturan, test koşullarının çevik karşılığı olarak kabul kriterleri
  - Kabul testine dayalı geliştirme (ATDD) Çevik metodolojilerde sıklıkla kullanılan, “önce test” yaklaşımını temel alan yüksek seviyeli bir test biçimi (sistem testi ve kabul testi) olarak.
- Bölümlerde açıklanan tekniklerin her biri belirli türdeki kusurları tespit etme konusunda özel bir amacı vardır. İşbirlikçi yaklaşımlar ise işbirliği ve iletişim yoluyla kusurlardan kaçınmaya da odaklanır.

4.5.1 İşbirlikçi Kullanıcı Hikayesi Yazma

- Çevik yazılım geliştirmede kullanıcı hikayesi, kullanıcı, sistem veya yazılımı satın alan kişi veya diğer herhangi bir paydaş için değer yaratacak işlevsel bir artışı temsil eder.
- Kullanıcı hikayeleri, gereksinimleri geliştiricilerin, test uzmanlarının ve iş temsilcilerinin bakış açısından yakalamak için yazılır.
- Sıralı SDLC modellerinde, belirli bir yazılım özelliği veya işlevine ilişkin bu paylaşılan vizyon, gereksinimler yazıldıktan sonra resmi incelemeler yoluyla elde edilir.
- Çevik yaklaşımlarda ise paylaşılan vizyon, gereksinimlerin yazılması sırasında sık sık resmi olmayan incelemeler yoluyla veya gereksinimlerin test uzmanları, analistler, kullanıcılar, geliştiriciler ve diğer paydaşlar tarafından işbirlikçi bir şekilde birlikte yazılmasıyla elde edilir.
- Kullanıcı hikayeleri “3C” olarak bilinen üç unsurdan oluşur:
  - Card
  - Conversation
  - Confirmation

4.5.1 İşbirlikçi Kullanıcı Hikayesi Yazma

Card / Kart

- Bir kart, bir kullanıcı öyküsünü tanımlayan bir ortamdır (örneğin, bir scrum panosuna yerleştirilen bir kağıt parçası biçiminde fiziksel bir kart veya bir elektronik panodaki bir giriş olabilir).
- Kart; gereksinimi, bunun kritikliğini veya önceliğini, kısıtlamalarını, beklenen geliştirme ve test süresini ve en önemlisi hikaye için kabul kriterlerini tanımlar.
- Açıklama, ürün biriktirme listesinde kullanılacağı için doğru olmalıdır.
- Çevik ekipler kullanıcı hikayelerini çeşitli şekillerde belgeleyebilir. Popüler formatlardan biri:
  - Bir (amaçlanan kullanıcı) olarak şunu istiyorum (amaçlanan eylem), böylece (eylem amacı/sonucu, elde edilen kar), ardından kabul kriterleri gelir.
- Bu kriterler farklı şekillerde de belgelendirilebilir.
- Kullanıcı hikayelerinin belgelenmesinde benimsenen yaklaşım ne olursa olsun, belgelemenin hem kısa hem de onu uygulayacak ve test edecek ekip için yeterli olması gerekir.
- Kartlar müşteri gereksinimlerini belgelemekten ziyade temsil eder.
- Kart hikayenin metnini içerse de, ayrıntılar konuşma sırasında çözülür ve onaya kaydedilir.

4.5.1 İşbirlikçi Kullanıcı Hikayesi Yazma

Conversation / Konuşma

- Konuşma, yazılımın nasıl kullanılacağını açıklar.
- Konuşma belgelenebilir veya sözlü olabilir.
- Geliştiricilerden ve iş temsilcilerinden farklı bir bakış açısına sahip olan test uzmanları, düşünce, fikir ve deneyim alışverişine değerli katkılar sağlar.
- Konuşma, yayın planlama aşamasında başlar ve hikaye planlandığında devam eder.
- Ürüne ilişkin üç temel bakış açısına sahip paydaşlar arasında gerçekleştirilir:
  - müşteri/kullanıcı
  - Geliştirici
  - test uzmanı

4.5.1 İşbirlikçi Kullanıcı Hikayesi Yazma

Confirmation / Onay

- Onay, bir kullanıcının hikayesinin ayrıntılarını ileten ve belgeleyen ve hikayenin ne zaman tamamlanacağını belirlemek için kullanılabilen kapsam öğelerini temsil eden kabul kriterleri biçiminde gelir.
- Kabul kriterleri genellikle bir görüşmenin sonucudur.
- Kabul kriterleri, test uzmanının hikayenin eksiksizliğini doğrulamak için kontrol etmesi gereken test koşulları olarak görülebilir.
- Kullanıcı hikayeleri hem işlevsel hem de işlevsel olmayan özellikleri ele almalıdır.
- Tipik olarak, test uzmanının benzersiz bakış açısı, eksik ayrıntıları veya işlevsel olmayan gereksinimleri belirleyerek kullanıcı öyküsünü iyileştirecektir.
- Test uzmanı, işletme temsilcilerine kullanıcı hikayesi hakkında açık uçlu sorular sorarak, bunu test etme yolları önererek ve kabul kriterlerini onaylayarak girdi sağlayabilir.
- Kullanıcı hikayelerinin ortak yazarlığı, beyin fırtınası veya zihin haritalaması gibi teknikleri kullanabilir.

4.5.1 İşbirlikçi Kullanıcı Hikayesi Yazma

Confirmation / Onay

- İyi kullanıcı hikayeleri INVEST adı verilen özellikleri karşılar, yani bunlar şunlardır:

  - **Independent/Bağımsız**—birbirleriyle örtüşmezler ve herhangi bir sırayla geliştirilebilirler.
  - **Negotiable/Pazarlığa açık**—bunlar açık özellik sözleşmeleri değildir; bunun yerine ayrıntılar, geliştirme sırasında müşteri ve geliştirici tarafından birlikte oluşturulacaktır.
  - **Valuable/Değerli**—bir kez uygulandıklarında müşteriye katma değer sağlamalıdırlar.
  - **Estimable/Tahmin edilebilir**—müşteri bunlara öncelik verebilmeli ve ekip, projeyi daha kolay yönetmek ve ekibin çabalarını optimize etmek için tamamlanma sürelerini tahmin edebilmelidir.
  - **Small/Küçük**—bu, ekibin kapsamlarını anlamasını kolaylaştırır ve hikayelerin oluşturulmasını daha kolay yönetilebilir hale getirir.
  - **Testable/Test edilebilir**—bu, iyi bir gereksinimin genel özelliğidir; Hikaye uygulamasının doğruluğu kolayca doğrulanabilir olmalıdır.

4.5.2 Kabul Kriterleri

- Kabul kriterleri, bir ürünün (kullanıcı hikayesi veya bu kabul kriterlerinin bir parçası olduğu Ürün İş Listesi Öğesi tarafından tanımlandığı ölçüde) müşteri tarafından kabul edilebilmesi için karşılaması gereken koşullardır.
- Bu açıdan bakıldığında kabul kriterleri, kabul testleri tarafından kontrol edilmesi gereken test koşulları veya kapsam maddeleri olarak görülebilir.
- Kabul kriterleri aşağıdakiler için kullanılır:
  - Kullanıcı hikayesinin sınırlarını tanımlamak için
  - Geliştirici ekibi ile müşteri arasında fikir birliğine varmak için
  - Hem olumlu hem de olumsuz test senaryolarını tanımlamak için
  - Kullanıcı hikayesi kabul testinin temeli olarak
  - Bir Doğru planlama ve tahmin için araç

4.5.3 Kabul Testine Dayalı Geliştirme (ATDD)

- Kabul Testi Odaklı Geliştirme (ATDD), önce test yaklaşımıdır.
- Kullanıcı hikayesi uygulanmadan önce test senaryoları oluşturulur.
- Test senaryoları, müşteriler, geliştiriciler ve test uzmanları gibi ürüne farklı bakış açılarına sahip ekip üyeleri tarafından oluşturulur.
- Test senaryoları manuel veya otomatik olabilir.
- İlk adım, ekip üyelerinin kullanıcı öyküsünü ve kabul kriterlerini analiz ettiği, tartıştığı ve yazdığı spesifikasyon çalıştayıdır.
- Bu süreçte hikayedeki eksiklikler, belirsizlikler, çelişkiler veya diğer türlü kusurlar gibi her türlü sorun giderilir.
- Bir sonraki adım testler oluşturmaktır.
- Bu, bir ekip tarafından toplu olarak veya bir test uzmanı tarafından bireysel olarak yapılabilir.
- Her iki durumda da testlerin doğrulanması iş temsilcisi gibi bağımsız bir kişi tarafından gerçekleştirilir.
- Testler, kullanıcı hikayesinin belirli özelliklerini tanımlayan, kabul kriterlerine dayanan örneklerdir.
- Bu örnekler ekibin kullanıcı hikayesini doğru şekilde uygulamasına yardımcı olur.

4.5.3 Kabul Testine Dayalı Geliştirme (ATDD)

Örnek

- Bir ekibin aşağıdaki kullanıcı hikayesini test etmeyi planladığını varsayalım.
- Kullanıcı hikayesi US-002-02
- Oturum açmış bir banka müşterisi olarak başka bir hesaba para aktarabilmek istiyorum.
- Kabul kriterleri
  - (AC1) transfer tutarı hesap bakiyesini aşamaz
  - (AC2) hedef hesap doğru olmalıdır
  - (AC3) geçerli veriler için (tutar, hesap numaraları) kaynak hesap bakiyesi transfer miktarı kadar azalır ve hedef hesap bakiyesi artar
  - (AC4) transfer tutarı pozitif olmalı ve doğru para miktarını temsil eder (yani maksimum iki ondalık basamağa kadar doğru olmalıdır)
- Testlerin kabul kriterlerinin karşılandığını doğrulaması gerektiğini akılda tutarak, buradaki pozitif fonksiyonel testlerin örnekleri ile ilgili aşağıdaki testler olabilir:

4.5.3 Kabul Testine Dayalı Geliştirme (ATDD)

Örnek

- TC1: "tipik" bir transfer yapın; örneğin, bakiyesi 2735,45$ olan bir hesaptan başka bir doğru hesaba 1500$ tutarında transfer yapın; beklenen sonuç: hesap bakiyesi = 1235,45 ABD dolar azalır, hedef hesabın bakiyesi 1500 ABD dolar artar
- TC2: hesap bakiyesinin tamamının başka bir doğru hesaba doğru şekilde aktarılmasını sağlayın (örneğin, bakiyesi 21,37 ABD doları olan bir hesaptan diğerine 21,37 ABD doları tutarında transfer); beklenen sonuç: hesap bakiyesi = 0 $, hedef hesap bakiyesi 21,37 $ artar

- İkinci test senaryosu, hesap bakiyesi ile transfer tutarı arasındaki fark için sınır değer analizini kullanır.
- Bir sonraki adım, testlerin kabul kriterlerinin karşılandığını doğrulamak için olduğunu akılda tutarak negatif testler oluşturmaktır. Örneğin, bir test uzmanı aşağıdaki durumları dikkate alabilir:

4.5.3 Kabul Testine Dayalı Geliştirme (ATDD)

Örnek

- TC3: transfer tutarı kaynak hesabın bakiyesinden fazla olduğunda başka bir doğru hesaba transfer yapmaya çalışın (AC1'in doğrulanması)
- TC4: doğru transfer tutarı ve bakiyeyle transfer yapmaya çalışın, ancak mevcut olmayan hesap (AC2'nin doğrulanması)
- TC5: aynı hesaba doğru transfer tutarı ve bakiyeyle transfer yapmaya çalışın (AC2'nin doğrulanması)
- TC6: yanlış miktarda transfer yapmaya çalışın (AC4'ün doğrulanması)

- Elbette, bu testlerin her biri için test uzmanı, belirli program davranışını doğrulamak amacıyla bir dizi test verisi tanımlayabilir.
- Bu test senaryolarını oluştururken test uzmanı kara kutu tekniklerini (örneğin eşdeğerlik bölümleme veya sınır değer analizi) kullanabilir.