YAZILIM GEÇERLEME VE SINAMA

Yazılım Testine Giriş

Test Nedir?

Yazılım Testi:

Yazılım testi, bir uygulamanın yaşam döngüsünde kritik bir adımdır. Testin genel tanımı, bir bireyin ya da topluluğun yeteneklerini, bilgilerini ve becerilerini değerlendirmek için yapılan sınamadır.

Test Nedir?

Bilişim sektöründe ise bu tanım, ürün kalitesini değerlendirmek, üzerindeki kusurları ve problemleri belirleyerek bu kusurları gidermek amacıyla yapılan bir sınamaya işaret eder.

Test Aktivitesinin Amacı: Testin temel amacı, bir yazılım ürününün kalitesini belirlemek ve potansiyel kusurları tespit etmektir.

Temel Kavramlar

Kalite: Bir yazılım ürününün belirtilen gereksinimleri karşılaması, kullanıcı beklentilerini yerine getirmesi ve belirli standartlarda performans göstermesi durumudur.

Kalite Nedir?

"Kalite" kelimesi Fransızcadan dilimize geçmiştir ve **"nitelik"** anlamına gelir. Bu, somut olmayan bir kavram olmakla birlikte, genel olarak bir hizmet ya da ürünün belirlenen ihtiyaçlara ne derece cevap verdiğini ifade eden özellikleri ve yetenekleri barındırmasıdır.

Kalite Nedir?

Bazı tanımlara göre kalite, bir şeyin amaçlanan kullanım için ne kadar uygun olduğu, belirtilen gereklilikleri ne ölçüde karşıladığı ya da bir nesnenin doğasındaki niteliklerin bu gereksinimleri ne kadar karşıladığı olarak ifade edilir.

Yazılım Kalitesi

Yazılım kalitesi ise IEEE tanımına göre;

1. Bir sistemin, bileşenin ya da bir sürecin belirtilen şartları karşılama derecesi.
2. Bir sistemin, bileşenin ya da sürecin müşteri ya da kullanıcı beklenti veya gereksinimlerini karşılama derecesidir.

Yazılım Kalitesi

Bu iki tanım, modern kalite güvencesi kurucuları **Philip B. Crosby** ve **Joseph M. Juran** tarafından düzenlenen, yazılım kalitesinin iki alternatif tanımını sunar:

1. Kalite gereksinimlere uygunluk anlamına gelir.
2. Kalite, müşterilerin ihtiyaçlarını karşılayan ve böylece ürün memnuniyeti sağlayan ürün özelliklerinden oluşur. Kalite eksikliklerden kurtulmaktan ibarettir.

Yazılım Kalitesi

Bir diğer tanım da **Roger S. Pressman**'dan; yazılım kalitesi açıkça belirtilen işlevsel ve performans gereksinimlerine, açıkça belgelendirilmiş geliştirme standartlarına ve profesyonelce geliştirilmiş yazılımlardan beklenen kapalı özelliklere uygunluk demektir.

Tanım ve Kavramlar

Kusur (Defect): Yazılımda mevcut olan ve yazılımın istenen şekilde çalışmasını engelleyen veya kısıtlayan problem ya da hata.

Bu temel kavramların anlaşılması, yazılım test sürecinin ve öneminin daha iyi kavranmasına yardımcı olacaktır.

Kusur(Defect)

Kusur, temelde bir beklenti veya gereksinimden sapma olarak tanımlanır. Detaylı bir tanımlama ile; bir öğenin istenilen nitelikleri **eksiksiz, etkili** ve **güvenli** bir şekilde taşımasına engel olan veya bu öğenin belirlenen amacını yerine getirmesini engelleyen *kırılganlık veya eksikliktir.*

Kusur(Defect)

Özellikle yazılım dünyasında bir kusur, bir yazılımın belirlenen gereksinimleri veya kullanıcının beklentilerini yerine getirememesi durumunu ifade eder.

Kusur(Defect)

Bu, yazılımın hatalı sonuçlar üretmesine veya beklenmeyen bir şekilde çalışmasına yol açan kodlama veya mantıksal bir hata olabilir. Bu kusurlar, yazılımın işleyişinde, performansında, donanım uyumluluğunda ya da genel işlevselliğinde karşımıza çıkabilir.

Kusur(Defect)

IEEE Standartları: Yazılım testi ile ilgili pek çok konu, IEEE 610 ve IEEE 982 standartları altında detaylıca incelenmiştir. Özellikle "defect" kavramının açıkça tanımlanması amacıyla IEEE 610.12-90 standardı hazırlanmıştır. Bu standarda göre:

- **Hata (error):** Hesaplanan ve gerçek sonuç arasındaki farkı ifade eder.

Kusur(Defect)

- **Yanlışlık, Hata, Kusur (fault):** Bir programdaki yanlış adımı, işlemi veya veri tanımını belirtir.
- **Başarısızlık (failure):** Bir hatanın olumsuz sonuçlarıdır.
- **Hata (mistake):** Hatalı sonuçları olan bir insan eylemidir. Genel kullanımda "defect" kelimesi, genellikle "fault" anlamında kullanılır.

Kalite ve Kusur (Özet)

Kalite: Müşterinin tatminini sağlayan bir ürün ya da hizmetin özelliklerinin bütünüdür.

Defect: Müşteri ihtiyaçlarının karşılanamamasına neden olan, belirlenen gereksinimlerden ya da beklentilerden sapma olarak tanımlanabilir.

Kalite ve Kusur (Özet)

Bu kavramların anlaşılması, yazılım mühendisliği alanında ürün kalitesini artırmak ve kusurları minimize etmek için kritik öneme sahiptir.

Giriş: Yazılım Testi Nedir?

Yazılım testi, girdileri alıp beklenen çıktılarla karşılaştırma yaparak yazılımın kalitesini ve işlevselliğini değerlendirme sürecidir. Bu, yazılımın teknik, işlevsel, fonksiyonel ve kullanıcı gereksinimlerine ne kadar uygun olduğunu belirlemek için gerçekleştirilen kritik bir adımdır. Yazılım testi, yazılımın doğru ve beklenen şekilde çalışıp çalışmadığını doğrulama işlemlerini içeren kapsamlı bir süreçtir.

Uluslararası Standartlar ve SWEBOK

IEEE tarafından oluşturulan ve uluslararası bir standart olan "Software Engineering Body of Knowledge - SWEBOK" kılavuzunda yazılım testine dair bir tanım yapılmıştır.

SWEBOK®

SWEBOK

Yazılım testi, genellikle sonsuz olan uygulama alanlarından özenle seçilmiş bir dizi test durumunda bir yazılımın davranışının, beklenen sonuçlara karşı dinamik bir şekilde doğrulanmasıdır.

Kısaca belirli bir yazılım ürününün, potansiyel olarak sınırsız çalışma alanından ***stratejik*** olarak seçilen davranışların, ***sınırlı*** test senaryoları üzerinden ***beklenen*** sonuçlarla ***dinamik*** bir şekilde karşılaştırılması sürecini ifade eder. Bu süreçte öne çıkan temel kavramları şu şekilde inceleyebiliriz:

1. Stratejik Seçim: Bu, risk analizlerinin yapılarak, potansiyel sorunların belirlenip, en etkili test senaryolarının seçilmesi sürecini ifade eder. En etkili seçimin yapılabilmesi için risk analizi tekniklerinin ve test mühendisliği uzmanlığının kullanılması gerekmektedir.

2. Sınırlı Test Sayısı: Her ne kadar bir yazılımda teorik olarak sınırsız miktarda test senaryosu oluşturulabilecek olsa da, kapsamlı testlerin süresi uzun olabilir. Kaynakların sınırlı olması nedeniyle, kritiklik seviyeleri belirlenerek en uygun test senaryoları seçilmeli ve optimal bir denge sağlanmalıdır.

SWEBOK

3. Beklenen Sonuçlar: Test edilen yazılımdan alınan sonuçların, önceden belirlenmiş gereksinimlere ve beklentilere uygun olup olmadığının kontrol edilmesi gerekmektedir.

4. Dinamik Doğrulama: Yazılımın gerçekte nasıl performans gösterdiğini görmek için, canlı ortamlarda çalıştırılarak test edilmesi gerekmektedir. Yazılımın türü ve özelliklerine bağlı olarak farklı test yöntemleri kullanılabilir.

Yazılım Testinin 3 Temel Kuralı

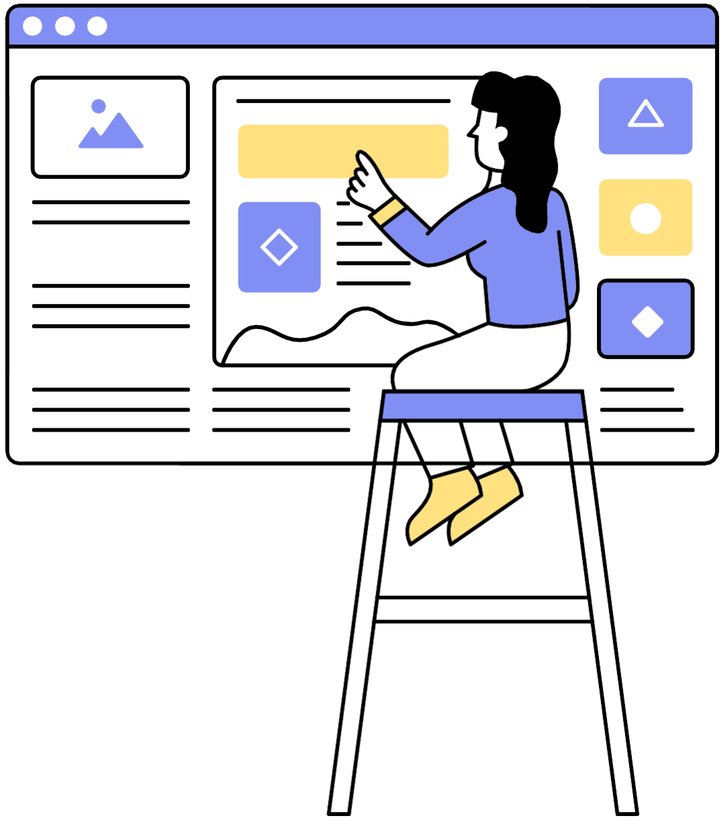
1. Testin amacı kusurları / defect'leri tespit etmektir.
2. Test sadece beklenen senaryoları değil (sunny day scenarios) aynı zamanda beklenmeyen senaryoları da (rainy day scenarios) kapsamalıdır.
3. Test gereksinimlerle birlikte başlar. Yani başta senaryolara bakılarak gereksinimlerin ne olduğu öğrenilmelidir ve kullanıcı perspektifinin sağlanmasıyla defect'ler bulunmalıdır.

Testi Kim Yapar?

IT sektöründe, özellikle kurumsal şirketlerde, yazılım gereksinimlerini analiz eden özel ekipler bulunmaktadır. Geliştiriciler, "birim testi" olarak bilinen test süreçlerini gerçekleştirirler. Son kullanıcının da katıldığı "kullanıcı kabul testi" ile ürün değerlendirilir. Bu test sürecine proje yöneticisi de dahil olabilir.

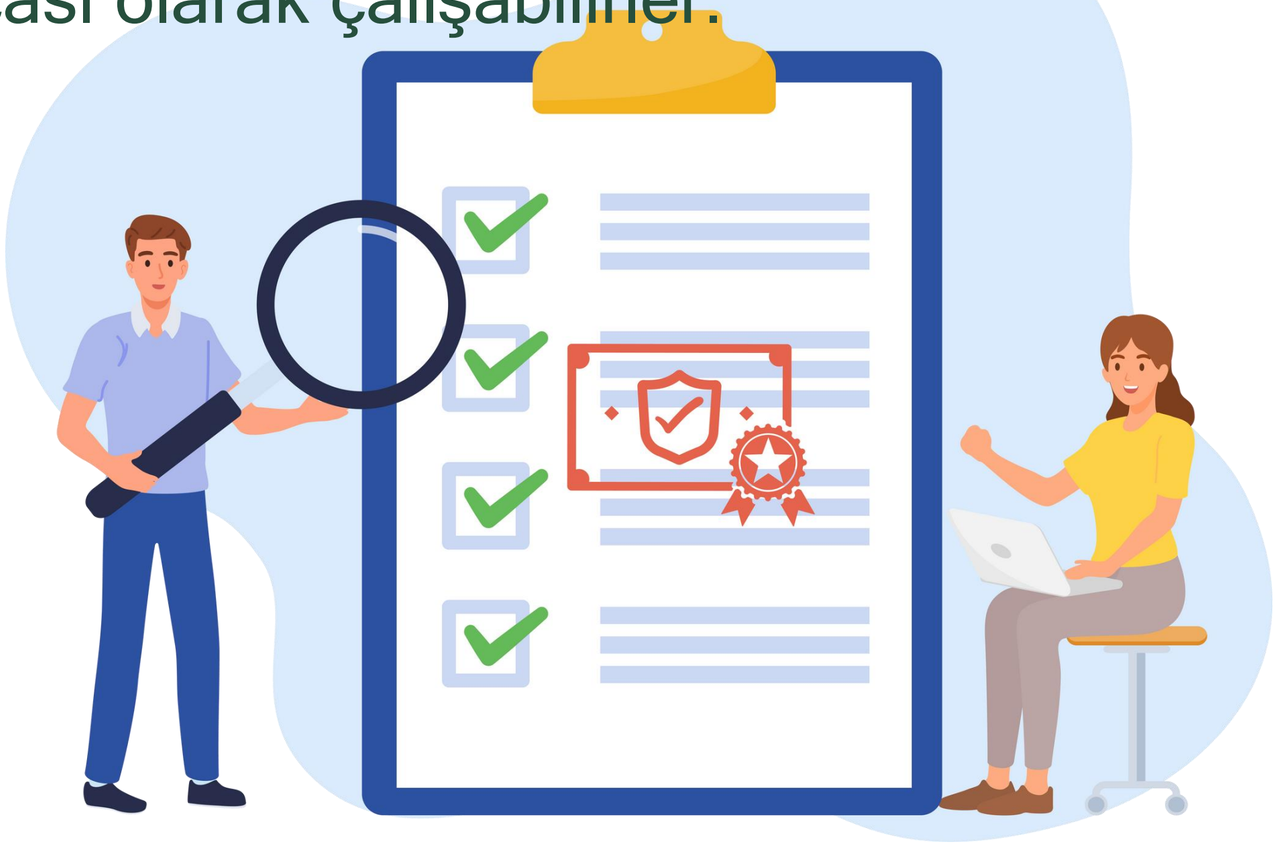
Yazılım Test Uzmanı / Mühendisi

Yazılım test mühendisleri, yazılımların belirlenen özellikleri ve performans kriterlerini karşılamasını doğrulamak için test prosedürleri oluşturur ve bunları uygular. Bunlar, yazılım geliştirme sürecinde, ürünlerin istenen standartlara uygun çalıştığından emin olmak için görevlendirilen profesyonellerdir.



Yazılım Test Uzmanı Çalışma Alanları

Kalite güvence (QA) veya kalite kontrol (QC) ekipleriyle birlikte veya bu ekiplerin parçası olarak çalışabilirler.



Yazılım Test Uzmanı Ana Faaliyetleri

1. Test faaliyetlerinin planlanması.
2. Test durumu ve hataların raporlanması.
3. Müşteriyle etkileşim (örn. kullanıcı kabul testleri).
4. Test gereklilik belgelerinin incelenmesi.
5. Test senaryolarının oluşturulması.
6. Testlerin yürütülmesi, kusurların tespiti ve önceliklendirilmesi.
7. Test ve proje süreçlerinin iyileştirilmesi.

Yazılım Testi Süreci Ne Zaman Durdurulur?

Testin ne zaman sonlandırılacağını tespit etmek kolay bir süreç değildir. Testlerin tamamlandığından %100 emin olmak mümkün değildir! Yine de, test sürecini sonlandırmaya karar verirken şu kriterleri dikkate alabiliriz:

Yazılım Testi Süreci Ne Zaman Durdurulur?

1. Testin önceden belirlenen bitiş tarihine ulaşması.
2. Tüm test senaryolarının başarıyla tamamlanması.
3. Fonksiyonel ve kod bazlı bölümlerin belirli bir seviyeye kadar tamamlanması.
4. Hata ve kusur oranının kabul edilebilir bir seviyeye inmesi ve kritik öneme sahip kusurların bulunmaması.
5. Üst yönetimin alacağı karar.

Yazılım Testinin Önemi Nedir?

- Yazılım kusurları erken aşamada tespit edilmeli ve ürün müşteriye sunulmadan önce kalite kontrolü yapılmalıdır.
- Eksik veya yanlış tespit edilen bir kusur önemli işletme zararlarına yol açabilir.
- Test süreçleri, yazılımın daha stabil ve kullanıcı dostu olmasını sağlar.

Yazılım Testinin Önemi Nedir?

- Etkili bir test süreci, düşük maliyetle yüksek kaliteli bir ürün elde etmeyi mümkün kılar.
- Düzenli ve etkili testler, yazılımın uzun vadede daha az bakım gereksinimi ile çalışmasına olanak tanır.
- Hataların ve kusurların geliştirme sürecinin erken evrelerinde tespit edilmesi, ilerleyen aşamalarda daha yüksek maliyetlerin önüne geçer.

Yazılım Testinin Önemi Nedir?

- Piyasaya sürülen ürünlerde kullanıcı deneyimi kritik bir rol oynamaktadır. Yazılımın kullanıcı dostu ve anlaşılır olmasını garantileyen detaylı test süreçleri, müşteri memnuniyetini artırır. Bu nedenle, üstün bir kullanıcı deneyimi için kaliteli yazılım test hizmetlerine başvurmak esastır.

Hata Ayıklama Stratejileri

Bilgisayarlı sistemlerin en büyük sorunlarından biri hata durumunda hangi bileşenin kusurlu olduğunun belirlenmesidir. Kusur, donanımda olabileceği gibi yazılımda da olabilir. Kusurun yazılımda olduğunun bulunmasından sonra yazılımdaki hatanın nerede olduğunu bulmak yeni bir sorun olarak ortaya çıkar.

Hata Ayıklama Stratejileri

Test aşamasında herhangi bir hata bulunması başarı olarak değerlendirilir. Bulunan hataların giderilmesi için hata ayıklama süreci başlatılır. Test sırasında bulunan hataların nereden kaynaklandığının belirlenmesi genellikle geliştiricinin deneyimine bağlıdır. Çoğu zaman deneme-yanılma yöntemi kullanılarak hatanın giderilmesi bir rastlantıya bırakılır. Oysa bu iyi bir yöntem değildir.

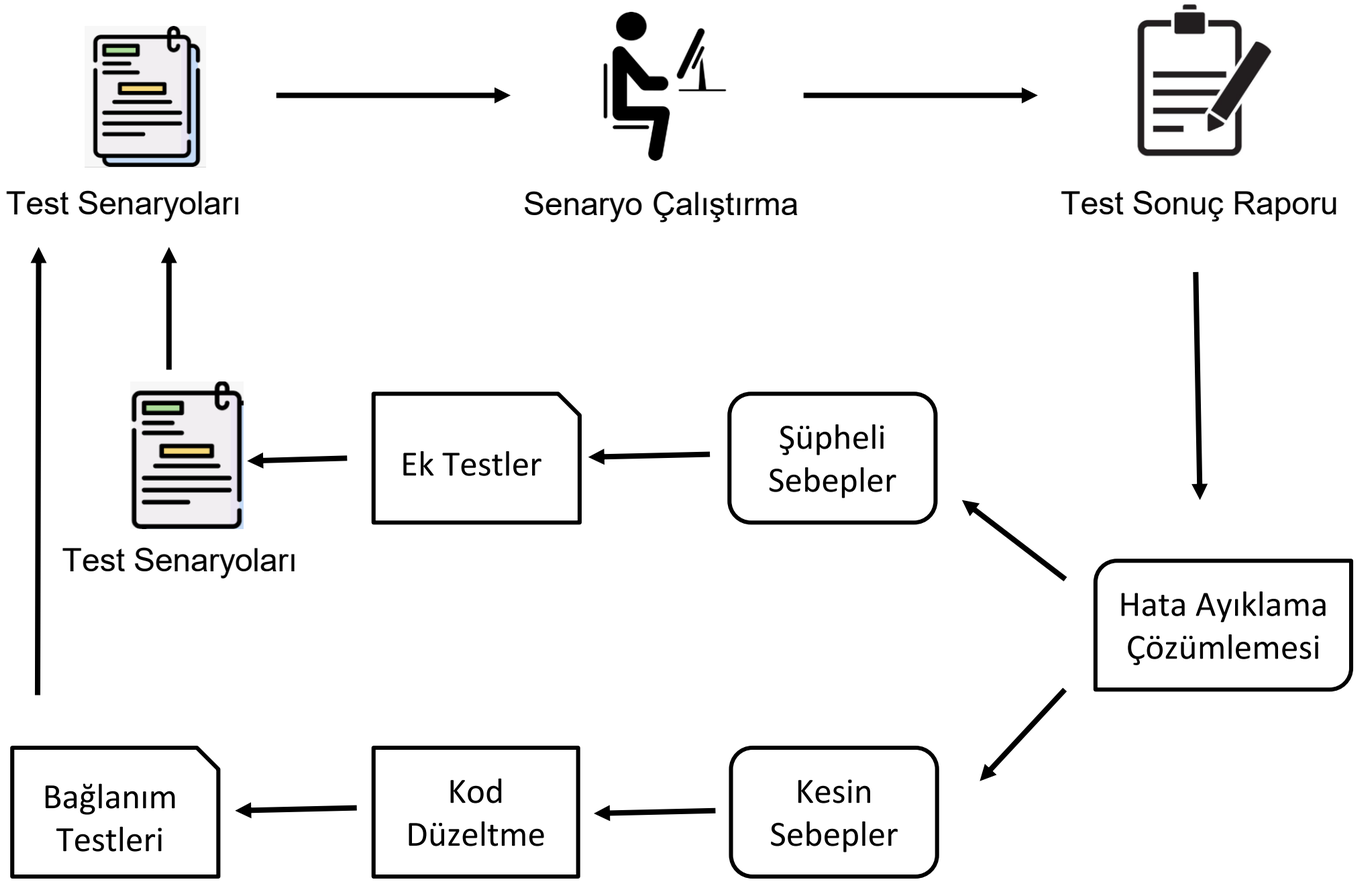
Hata Ayıklama Stratejileri

Hata ayıklama süreci normalde test aşamasının sonunda başlar. Testlerin mutlaka tek bir aşama şeklinde yapılma zorunluluğu olmadığı için, aslında test yapılabilecek her noktada hata ayıklama vardır. İlk kodlama anında ortaya çıkarılan hatalardan sistem kullanımı sırasında rapor edilen hatalara kadar her aşamada karşılaşılan yazılım hataları için birer hata ayıklama süreci uygulanır.

Hata Ayıklama Stratejileri

Hata ayıklama sürecinin amacı, yazılımın kusurlu çalışan kısmının neden kaynaklandığını bulmak ve gidermektir. Hataya neden olduğu kanısına varılan belirtilerin takip edilmesiyle ya başarıya ulaşılır, yani hatalı çalışan kısım bulunur ve düzeltilir ya da başarısız olur, yani hata bulunmaz. Başarısız olma durumunda, hata ayıklayıcı kişi veya kodlayıcı test senaryoları yazarak kodun belirli kısımlarını test eder, gerekirse tekrarlar halinde hatayı arar.

Hata Ayıklama Stratejileri



Hata Ayıklama Süreci

Hata Ayıklama Stratejileri

Bazen yazılım hataları çözülmesi zor olabilir ve geliştiricilerin zamanının büyük bir bölümünü alabilir. Gerçekten karmaşık hatalar için hata ayıklama bile günler alabilir. Ancak süreci önemli ölçüde hızlandırması gereken belirli hata ayıklama yöntemleri vardır.

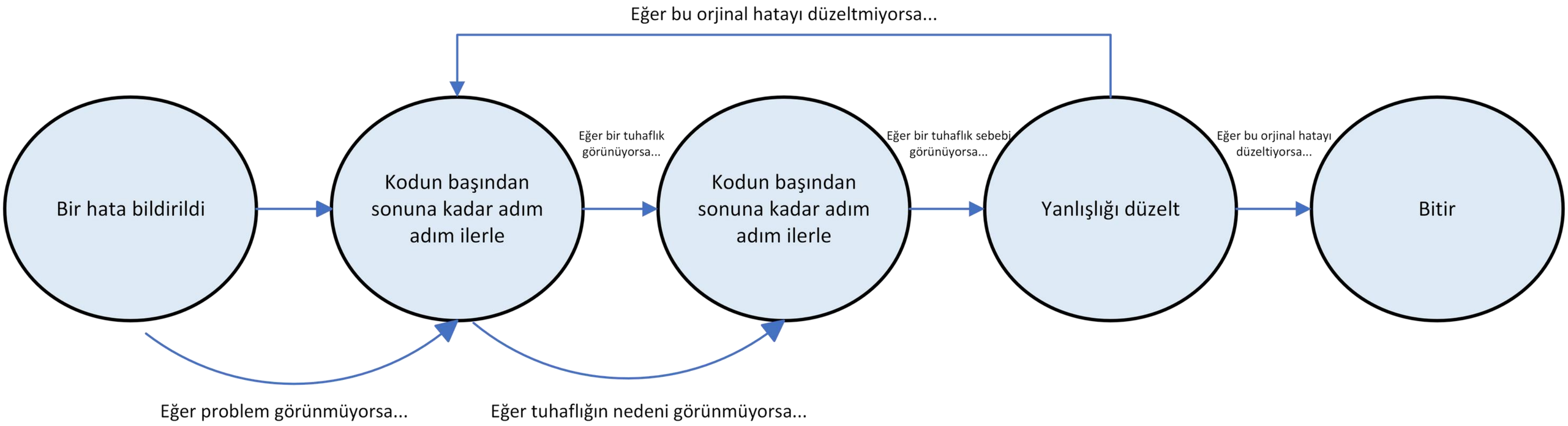
Hata Ayıklama Stratejileri

- **Brute force method** (Kaba kuvvet yöntemi)
- Rubber duck debugging (Lastik ördekle hata ayıklama)
- Bug clustering (Hata kümeleme)
- **Cause elimination method** (Nedeni ortadan kaldırma yöntemi)
- **Backtracking** (Geri izleme)
- Program slicing (Program dilimleme)
- Binary searching (İkili arama)
- Static analysis (Statik analiz)

Brute Force Yöntemi

- En yaygın hata ayıklama stratejisi muhtemelen kaba kuvvet yöntemidir; ancak aynı zamanda en az verimli olanıdır.
- Adından da anlaşılacağı gibi bu yöntem, yazılım hatalarını çözmeye yönelik organize ve yapılandırılmış bir yaklaşımdan yoksundur.
- Bunun yerine, geliştiriciler kodun üzerinden geçerek bir çözüme ulaşana kadar çeşitli düzeltmeler denerler.

Brute Force Yöntemi



Brute Force Yöntemi

- Tanımlanmış bir prosedür yoktur. Geliştiriciler kodla gelişigüzel bir şekilde ilgilenir ve uygulanabilir bir çözüm ortaya çıkarana kadar hataları çözerler.

- Bu koordinesiz hata ayıklamanın bir parçası olarak geliştiriciler genellikle aşağıdaki tekniklere başvurur:

Brute Force Yöntemi

Kaba Kuvvet hata ayıklama uygulamaları

Bellek Dökümü
Yazılım tanılama bilgilerini toplamak için dökümlerin incelenmesi.

İfadeleri Yazdır
Program yürütülürken çıktı çıktısının (işlevler, değerler) incelenmesi.

Hata ayıklayıcı kullanma
Kusurları çözmek için özel bir hata ayıklama aracının kullanılması.

Brute Force Yöntemi

- Bellek dökümleri, sistem durumları ve bellek içerikleriyle ilgili verileri içerir; bu, hataların teşhisinde son derece faydalıdır.
- Biraz daha karmaşık bir teknik, basılı ifadeleri dağıtmaktır . Bellek dökümlerinin aksine, yazdırma ifadeleri programın dinamiklerini görüntüler ve kolayca okunur.

Brute Force Yöntemi

- Son olarak, kaba kuvvet hata ayıklamasına hata ayıklayıcılar eşlik eder . Bu araçlar son derece yararlı olsa da, onlara çok fazla güvenmemek önemlidir.
- Yararlı olmalarına rağmen, hata ayıklamanın gerektirdiği eleştirel düşünmenin yerini almazlar.

Cause elimination Yöntemi

- Neden giderme yöntemi (Cause elimination), yazılım hatalarını tespit etmek ve çözmek için tümdengelimli bir yaklaşımı benimseyen bilimsel bir hata ayıklama stratejisidir.

Cause elimination Yöntemi

Neden Giderme Yönteminin Adımları:

- **Olası Nedenlerin Listesini Oluşturma:** İlk adımda, kusurun olası tüm nedenlerinin bir listesini oluşturun. Bu nedenlerin tamamının rasyonelleştirilmesine gerek yoktur. Bu sadece durumu değerlendirmeye yardımcı olacak varsayımlardır.

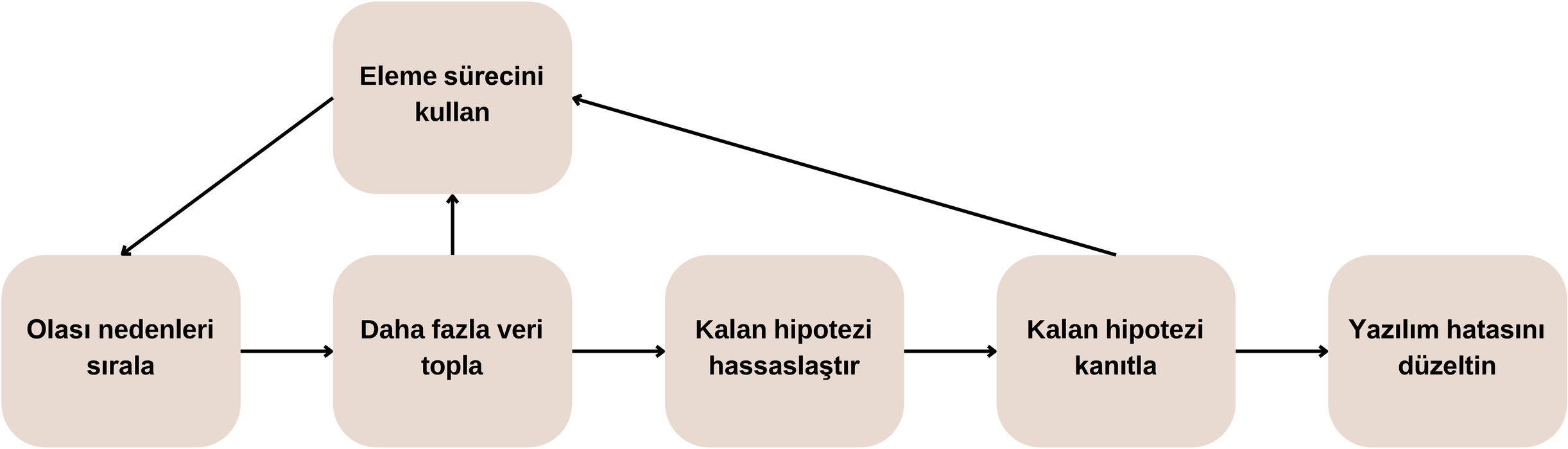
Cause elimination Yöntemi

- **Hipotezleri Test Etme:** Oluşturduğunuz her bir hipotezi tek tek test edin. Kod tabanınızı analiz ederken, ilerledikçe olası nedenleri bir bir ortadan kaldırın. Bu sürecin sonunda tek bir olası nedenle baş başa kalmalısınız.

Cause elimination Yöntemi

- **Teoriyi Geliştirme:** Eğer geriye kalan teori belirsizse veya tam olarak kanıtlanmamışsa, bu teoriyi daha da geliştirmeniz gerekmektedir.
- **Hatanın Çözümü:** Son adımda, geliştirdiğiniz teoriyi kullanarak yazılım hatasını çözmelisiniz.

Cause elimination Yöntemi



Backtracking Yöntemi

Küçük bir kod tabanı üzerinde çalışırken, hataları ayıklama yöntemi olarak geri izleme (Backtracking) idealdir. Hata ayıklamaya, programın yanlış sonucu verdiği noktadan başlamalı ve önceki adımları zihinsel olarak geri izleyerek yazılımın nerede yanlış çalışmaya başladığını tespit etmelisiniz.

Backtracking Yöntemi

Geri izleme, kodun beklenen bir durumdan yanlış sonuca nasıl geçtiğini anlamanızı sağlar. Bu, hatayı daha kolay çözmeniz için gereken bilgileri size sunar.

Ancak, bu yöntemin özellikle daha az kodun bulunduğu küçük kod tabanlarında daha etkili olduğunu unutmamak gerekir. Bu yaklaşım, hatanın tam olarak hangi kod satırları arasında olduğunu belirlemenizi sağlar, böylece hızla çözüme kavuşturabilirsiniz.

ÖZET

- Yazılım testi, yazılımın belirlenen spesifikasyonlara uygun şekilde çalıştığını garanti altına almak için esastır.
- Hata giderme stratejileri arasında Brute Force, Backtracking ve Cause Elimination bulunmaktadır.
- Brute Force, potansiyel hataların belirlenip düzeltilmesi için kör kuvvet yaklaşımını ifade eder.

ÖZET

- Backtracking, bir hatanın kökenini bulmak için yazılımın durumunu geriye doğru izlemeyi içerir.
- Cause Elimination, mümkün nedenleri bilimsel bir yaklaşımla ele alarak hataların nedenini bulmayı hedefler.
- Nedeni belirlenemeyen bir hatayla karşılaşıldığında, kod tabanını yeniden incelemek ve süreci baştan başlatmak gerekebilir.

ÖZET

- Geri izleme yöntemi, özellikle küçük kod tabanlarında hatayı bulmak için en etkilidir.
- Yazılımın yanlış çalıştığı bir noktadan başlayarak geriye doğru izlenmesi, hatalı kısmın belirlenmesine yardımcı olur.
- Test süreçleri, test faaliyetlerini tanımlama, raporlama ve kullanıcı kabul testleri gibi adımları içerebilir.

ÖZET

- Yazılım testi, kullanıcının deneyimini geliştirme ve yazılımın maliyetini optimize etme açısından kritik bir rol oynar.

ÖRNEK SORULAR

1. Yazılım testinin ana amacı nedir?
2. Brute Force hata giderme stratejisinin temel özelliği nedir?
3. Backtracking stratejisiyle hatanın kökenini nasıl bulabilirsiniz?
4. Bir hata veya problemi çözme yaklaşımı olarak Cause Elimination stratejisinin temel prensipleri nelerdir?
5. Küçük bir kod tabanında hangi hata giderme yöntemi en etkilidir ve neden?

ÖRNEK SORULAR

6. Hangi durumlarda yazılım test sürecini durdurmanız gerektiğini belirleyin.
7. Yazılımın doğru çalışmadığı bir yerden başlayarak, hangi strateji ile geriye doğru adımlarınızı izleyerek hatayı bulabilirsiniz?
8. Yazılım test mühendisinin ana faaliyetleri nelerdir? İki örnek verin.
9. Neden bir yazılımın geliştirmenin ilk evrelerinde bug'ların tespit edilmesi daha ekonomiktir?
10. Bir yazılımın kullanıcı deneyimini optimize etmek için hangi test süreci adımının kritik olduğunu açıklayın.