Yazılım Testi ve Proje Yönetimi

Turkish Testing Board https://www.turkishtestingboard.org/

International Software Testing Quality Board https://www.istqb.org/

ISQTB Glossary
https://glossary.istqb.org/en_US/search

15 Nisan 2024

Yazılım Testi ile Ne Yapılmaktadır?

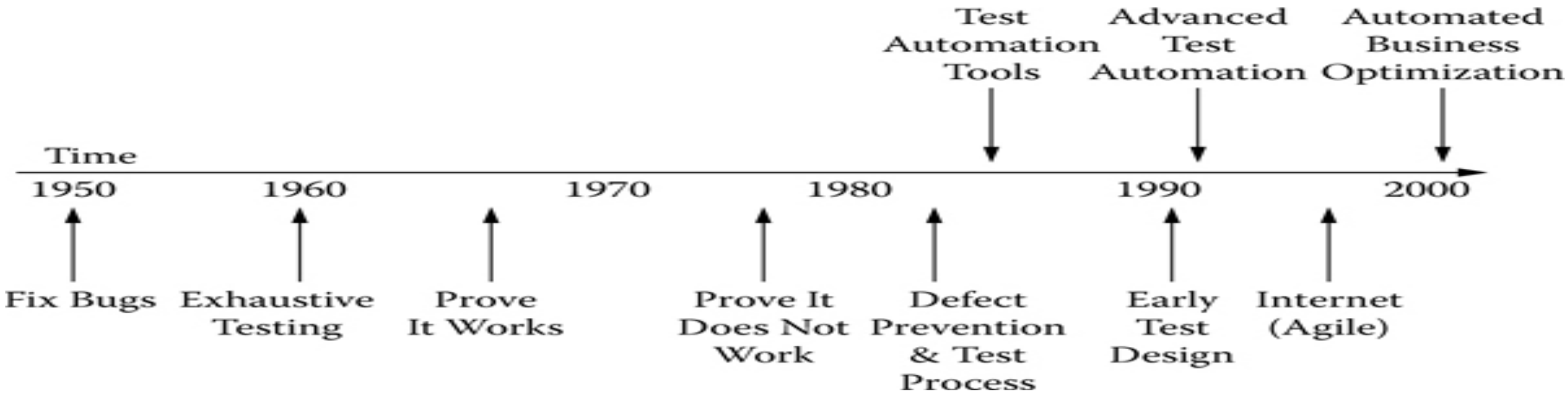
- Bilgi Teknolojileri alanında herhangi bir yazılım ürünü geliştirilirken yazılım testi sektörün bir alt disiplinine nasıl dönüşmüştür? sorusunun cevabı:
  - «High dependency on **reliable** software»

  olarak verilebilir.

- Yazılım Testinin temel ilkeleri nelerdir?
  - Bu disipline ilişkin kavram ve motivasyonlar şöyle özetlenebilir:
    - Quality Requirements (Nitelik Gereksinimleri)
    - Software is intangible (Yazılım soyuttur)
    - Faulty software is a serious problem (Hatalı yazılım önemli bir problemdir)
    - Testing helps to assess software quality (Test süreci ile yazılım kalitesinin değerlendirilmesine destek olunur.

Yazılımın Testi ve Yazılım Kalite Güvencesinin Tarihsel Gelişimi

Test Proseslerinin Gelişimi



Doğruluğun Kanıtlanması (Proof of Correctness)

- 60 ‘lı ve 70’lı yılların test yöntemidir.
- Bu yaklaşım günümüzde **kabul testi (acceptance test)** olarak uygulanmaktadır.
- Uygulamada zaman kaybı ve verimsiz (insufficient) bir yaklaşım olarak değerlendirilmesine rağmen bu düşüncenin zararları açıktır.
- Basit testler için, yazılımın **çalıştığını** ve teorik olarak **çalışacağını** kanıtlamak kolaydır.
  - Yazılımların çoğu bu yaklaşımı kullanılarak test edilmediğinden, gerçek uygulamanın implementasyonu (actual implementation) sırasında karşılaşılabilecek çok sayıda **defect (hata)** ortaya çıkması olasıdır.

Geleneksel /Traditional Testler ve Günümüz Testleri

- Geleneksel testler (1980'lerin başına kadar), günümüzün sistem testi idi.
  - Çalışan kodun müşteriye teslimi sonunda sisteme neler yapıldığı, yani sistemin neler yaptığı kontrol edilirdi.
- Günümüzde test süreci, test uzmanının yazılım geliştirme sırasında yaşam döngüsünün tümüne test aşamalarını teker teker eklemesidir.
  - *Kod testi* kodun yazımı tamamlandıktan sonra gerçekleşir.
    - ✓ Ama herhangi bir yanlışlık varsa, önceki geliştirme aşamalarına dönülmesi gerekir.
  - Hata (defect) bir tasarım belirsizliğinden veya bir programcıdan kaynaklanıyorsa, sorun oluşur oluşmaz bulmaya çalışılmalıdır.
    - ✓ Test için sürecin sonuçlanması asla beklenmemelidir.
- Hataların (defects) yarısının gereksinimlerin belirlenmesi aşamasında (yazılım ürününün ne yapmasını istiyoruz? Sorusu cevaplanırken) veya ürünün tasarımı aşamalarında oluştuğu görülmektedir.
  - Bu aşamalarda ortaya çıkan hatalar (defects) arttırıcı bir etkiyle kodlama sırasında daha fazla hata oluşturmaktadır.

Exhaustive (Kapsamlı) Test

- 60 ‘lı yıllarla birlikte ürünün kodlarındaki olası tüm yolların kapsamlı (exhaustive) testine önem verilir.
- Bu test ile, olası girdi /veri varyasyonlarının tümünün kullanılarak kodun kontrolü sağlandı .
- O yıllardaki kod yoğunluğunda bile bir uygulamayı tamamen test etmek imkansızdı.
  - Program girdilerinin etki alanı çok büyük olabilir
  - çok fazla olası giriş yolu olabilir.
- Tasarım ve spesifikasyon sorunlarının test edilmesinin zorluğu 60’ lı yıllarda anlaşılmıştı.
  - Bu nedenle de kapsamlı (exhaustive) testler o dönemde bile dikkate alınmadı ve teorik olarak olanaksız olduğu kanıtlandı.

80’li ve 90’lı yıllarda Yazılım Testi

- 80’li yıllar otomatik geliştirme aracı (tool) kullanılarak testin gerçekleştirildiği dönemlerdir.
- Geliştirilmesi hedeflenen uygulamanın **verimliliği** (efficiency) ve **kalitesinin** (quality) arttırılması hedeflenmeye başladı.
- Başlangıçta basit olan bu test kitleri, 90’lı yıllarla birlikte gelişti ve **early test design** kavramı doğdu.
- Yazılım Testi kavramsal olarak **testleri ve test ortamlarını planlama, tasarlama, oluşturma, sürdürme ve yürütme** olarak yeniden tanımlandı.
  - Bu, **yazılım testinin** bir **kalite güvence** bakış açısı olarak değerlendirilmesiydi.
  - Ayrıca, test edilebilirliğin **yaşam döngüsü** içerdiğinin tanımıydı.
- **Özetle:** iyi tanımlanmış bir test **yönetilen bir süreç** olarak ifade edildi.

21. Yüzyıla doğru: Agile (Çevik) Testler

- 90'lı yılların ortalarına kadar test kavramı **yazılım geliştirme yaşam döngüsüne ait** bir süreçti.
  - Diğer bir ifade ile, 90 lı yılların ortalarına kadar standart bir test modeli kullanılmadan yazılım geliştiriliyordu.
  - Bu da yazılım ürününün testini zorlaştırıyordu.
- **Çevik Test** kavramı ile test edilecek her şeyin önceden açık olarak tanımlanması gerekmediği fark edildi.
  - Gözden geçirilen her adımda beklenen sonucu tanımlamaya gerek yoktu, belgelerin gözden geçirilmesi yeterliydi.
- Çevik test tekniklerden bazıları, **keşifsel (exploratory)** testler, **hızlı (rapid)** testler ve **risk odaklı (risk based)** testlerdir.

Yeni Metodolojilerde Yazılım Testi

- Test, kısa sürümler içeren döngülere uygun olmalıdır.
  - Her bileşen, her yeni artış (increment) için anında tekrarlanabilen yeniden kullanılabilir test senaryoları gerektirir.
  - Aksi durumda artış gerçekleştikçe sistem güvenilirliğinin düşüşe geçmesi riski vardır.
    - ✓ Her artış, herhangi bir ek işlevi kapsayan yeni test senaryoları gerektirir.
- Testin çevik geliştirmeye uyarlanmasında test otomasyonu önemli bir araçtır.

Bir Vaka Çalışması Örneğinde
The risks of Using Faulty Software

- Her sistemin her bir sürümü, teslim edilmeden ve kullanıma sunulmadan önce uygun şekilde test edilmelidir.
- Bu, ürüne herhangi bir hasar vermeden önce hataların (defects) tespit edilerek giderilmesini sağlar.
- Sistem, bir siparişi hatalı yürüttüğünde müşteri, bayi ve üretici için ciddi mali sorunlar getirir ve üreticinin imajını da zedeler.
- Ortaya çıkmamış hatalar (*undiscoverde faults*), yazılımın çalışmasıyla ilgili riski artırır.

Test için Öncelikler

- Test süreci bir *spot-check yaklaşımıdır.*
  - Dinamik test ağırlıklı olarak gereksinimleri ne kadar gerçekleştirdiğini ölçer.
  - İlgili test nesnesine farklı test durumları (cases) farklı veri setleri için uygulanır.
- Test süreci yapılan testlerden daha fazlasını içerir.
  - «test planning, test analysis, the design, implementation of test cases».
  - «writing reports on test progress» ve «risk analysis».
  - Test aktiviteleri geliştirilen ürünün yaşam döngüsünden farklıdır ve test dokümantasyonu contract olarak adlandırılır.

Test Sınıflandırması

- Statik Testler
  - Yazılım Geliştirme Süreçlerinin testidir.
  - Gereksinimlerin, tasarımın ya da kodun implementasyondan önce test edilmesidir.
  - Belgelerdeki hatalar (faults) ne kadar erken farkedilir ve giderilirse, gelecekteki geliştirme süreci için o kadar doğru ve verimli gerçekleşecektir; böylece hatalı (flawed) kaynak materyal bulunmayacaktır.
- Dinamik Testler
  - Bir bileşen ya da sistemin çalıştırılarak test edilmesidir.

Statik Testin Dinamik Testten Farkı

- Statik test hataları (defects) önlemeyi amaçlarken, dinamik test hataların bulunması ve çözümlenmesi (finding and fixing) ile ilgilidir.
- Statik test verification / doğrulama sürecini gerçekleştirirken, dinamik test validation/sağlama sürecini yürütür.
- Statik test derlemeden önce gerçekleşirken, dinamik test derleme sonrasında gerçekleşir.

Verification /doğrulama : Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled (Gereksinimlerin yerine getirildiğinin deneme ve nesnel kanıtları ile doğrulanması.

Validation /sağlama : Confirmation by examination that a work product matches a stakeholder's needs (Ürününün paydaşın ihtiyaçlarını karşıladığının denemeller sonucunda sağlanması)

Sağlamanın /Validation Önemi

- Yazılım testi sadece sistemin müşteri gereksinimlerini, kullanıcı hikayelerini ya da diğer betimlemeleri kontrol etmesi değildir.
- Ürünün gerçek (real world) kullanıcılarının beklentilerini ne ölçüde karşıladığı da önemlidir.
- Sistemin amacına uygun kullanılmasının mümkün olup olmadığının kontrol edilmesi ve planlanan amacına ulaştığından emin olmak önemlidir.

Fault-Free Large System

- Böyle bir sistem yoktur.
  - Sistemin karmaşıklığı ve kod sayısına bağlı olarak değişir.
- «Many faults are caused by a failure to identify or test for exceptions during code development»
- Sistemin diğer parçaları mükemmel çalışsa bile, giriş verilerinin belli kısımlarında **faults** olabilir.
  - Sistem canlıya geçtiğinde büyük sistemlerde zaman zaman (giriş datasının belli kombinasyonları çalışırken) **failure** durumları ile karşılaşılabilir.

Freedom from Faults
Absence-of-errors is a fallacy

- «Hatalardan (faults) arınma test yoluyla elde edilemez»
  - Yapılan her bir test hatasız bile olsa, çok küçük programlar dışında, sonraki testlerin daha önceden mevcut olmayan hataların ortaya çıkmayacağı anlamına gelmez.
  - Test ederek hataların tamamen giderildiğini kanıtlamak mümkün değildir.

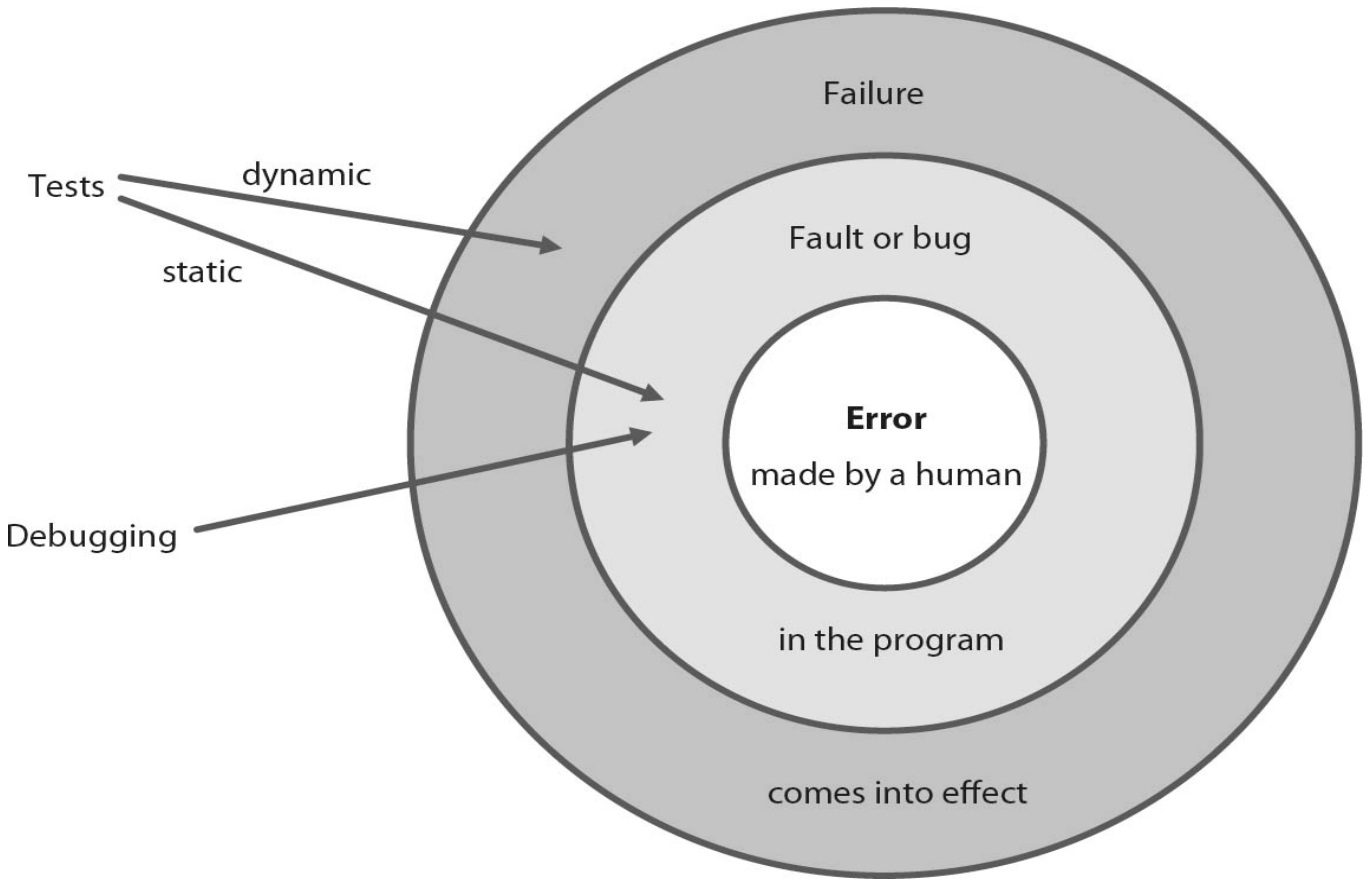
«Defect» ve «Fault» Terminolojisi

- The test basis as a starting point for testing
- What counts as a defect?
- Faults cause failures
- Defect masking
- People make errors
- False positive and false negative results
- Learning from mistakes

Fault /Kusur Nasıl Belirlenebilir?

- **Fault** durumu için geliştirilecek olanın ne olması gerektiği tam olarak önceden tanımlanmalıdır.
- Bu da test edilecek alt sistemin gerekliliklerinin bilinmesini gerektirir.
- Böylece herhangi bir işlev için **fault** durumun kararı hangi testlerin yapıldığına ve bunların test esasına göre verilecektir.

Fault – Error – Failure



Defect /Hata nedir?

- Sistemin davranışı ne zaman kullanıcının gerçek gereksinimleri ile uyuşmaz? sorusuna cevap aranır.
- Defect, tanımlanmış bir gereksinimin yerine getirilmemesi, çalışma zamanında veya test sırasında ile beklenen davranışta tutarsızlık olmasıdır.

- Yazılım sistemi sistemin eskiliği (yaşlanması) veya aşınma gibi nedenlerle failure /başarısız olmaz.
  - Herhangi bir hata /defect, yazılımın kodlanması ile ortaya çıkar, yalnızca sistemin çalışması sırasında görünür.

Fault, Failure ve Bug

- Sistemdeki **failure** durumunun nedeni, bir veya daha fazla **fault /kusur** olarak özetlenebilir.
  - Bu durum **defect /hata** olarak adlandırılır.

- Sistemdeki **failure** durumu test sırasında veya sistem çalıştırıldığında (run time) test uzmanı veya kullanıcı tarafından gerçekleşir.
  - Örneğin, sistem hatalı çıktı üretir veya çöker.

- **Bug** sözcüğü, kodda **error** vererek yanlış programlanmış veya unutulmuş bir komut gibi kodlama hatalarından kaynaklanan tanımlar.

Defect Masking

- Çözümün bir yerindeki fault /hata durumunun düzeltmesi, kodun başka yerlerinde beklenmedik yan etkilere neden olabilir.
- Çözümünü bir yerindeki herhangi bir fault durumu, ancak diğer fault durumlarının düzeltilmesi ile düzeltilebilir.
- Çözümün içerisindeki her fault sistemin çökmesine, yani bir failure neden olmaz.
  - Bazı failure durumları ile hiç karşılaşılmayabilir, bir kez oluşabilir ya da kullanıcıların tümünün maruz kaldığı bir durum olabilir.
  - Bazı failure durumları ile, neden oldukları yerde değil de ürünün çok farklı bir parçasında karşılaşılabilir.

Error /Kişisel Hata Nedenleri
Absences of Errors is Fallacy

- İnsan faktörü
- Zaman baskısı
- İşin karmaşıklığı
- Projenin katılımcıları arasındaki anlaşmazlık
- Sistem ile içsel (internal) ve dışsal (external) arayüzler arasındaki uyumsuzluk
- Proje katılımcılarının yeni teknolojiyi öğrenmedeki sıkıntısı
- Proje katılımcıları arasındaki anlaşmazlık

Debugging

- Testing is not debugging
  - Debugging, yazılım faults /kusurlarını hatalarını tespit ederken, fault oluşturan etkiyi ortaya çıkarmak için test işlemi gerçekleştirilir.
  - Faults bulma ve düzeltme sürecine hata ayıklama **debugging** denir .
  - Debugging süreci geliştiricinin sorumluluğundadır.