BSM462
Yazılım Testi

Hafta - 10

Yazılım Güvenilirliği

Dr. Öğr. Üyesi M. Fatih ADAK

fatihadak@sakarya.edu.tr

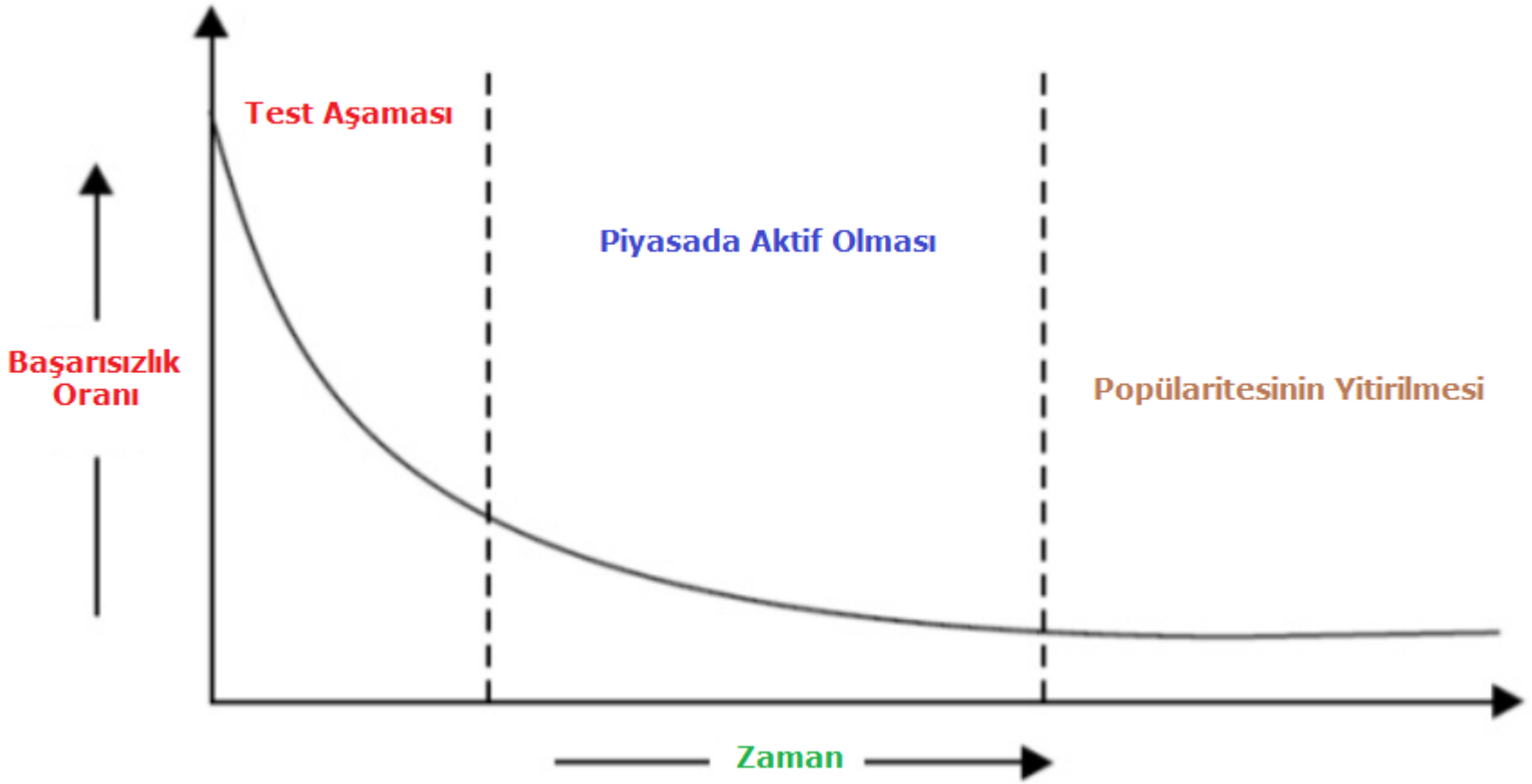
İçerik

- Tanım
- Yazılım güvenirliğinde anahtar kelimeler
  - Zaman
  - Zaman aralığı
- Yazılım güvenirliğine iki farklı yaklaşım
- Güvenirliği ölçmek
- Kümülatif başarısızlık sayısı
- Yazılım güvenirliğinin uygulamaları
- Operasyonal profiller (OP)
- Operasyonal profillerin kullanımı
- Güvenirlilik modellerinin geliştirilmesi
  - Temel model
  - Logaritmik model

Tanım

- Yazılım güvenilirliği yazılım kalitesinde önemli metriklerden biridir.
- Kullanıcı tabanlı kalite faktörü ve sistem işleyişi ile ilgilidir.
  - Kullanıcılar tarafından sürekli geri bildirimde bulunulan bir yazılımın güvenilirliği düşüktür.
- Hatasız bir yazılım yüksek güvenilir olarak kabul edilir.
  - Fakat hatasız bir yazılım oldukça zor bir süreçtir.
  - Doğru olmayan bir yazılımda bile eğer çok nadir çökme yaşanıyorsa güvenilir olarak kabul edilebilir.

Yazılım Güvenirliği Süreç Grafiği



Yazılım Güvenilirliğinde Anahtar Kelimeler

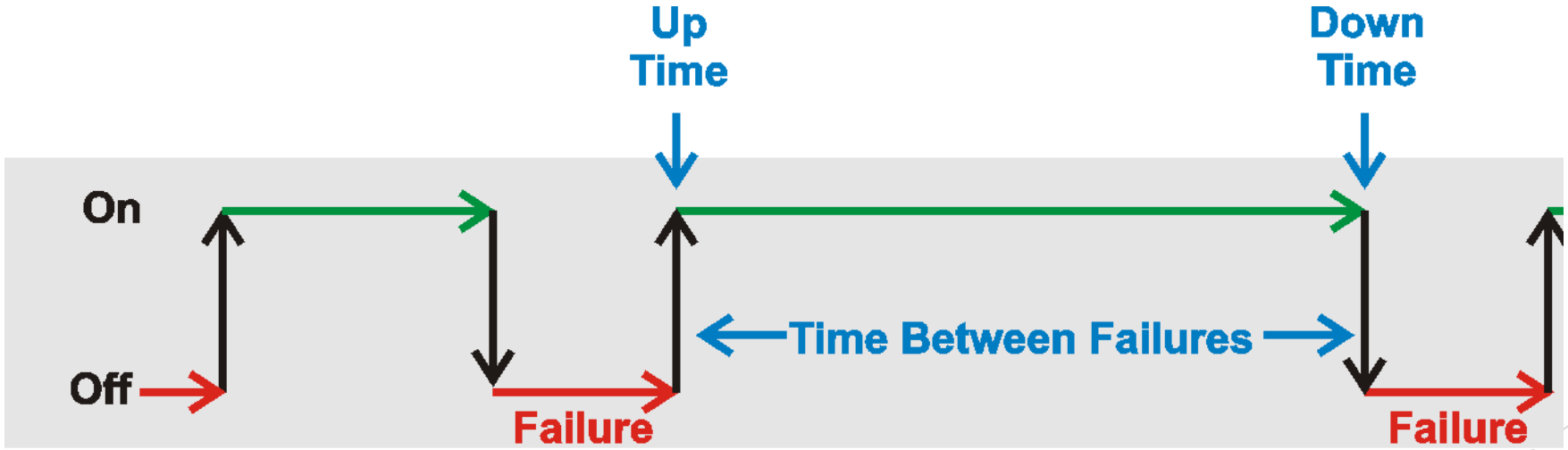
- Fault (Arıza)
- Failure (Başarısızlık)
- Time (Zaman)
- Time Interval (Zaman Aralığı)

Time (Zaman)

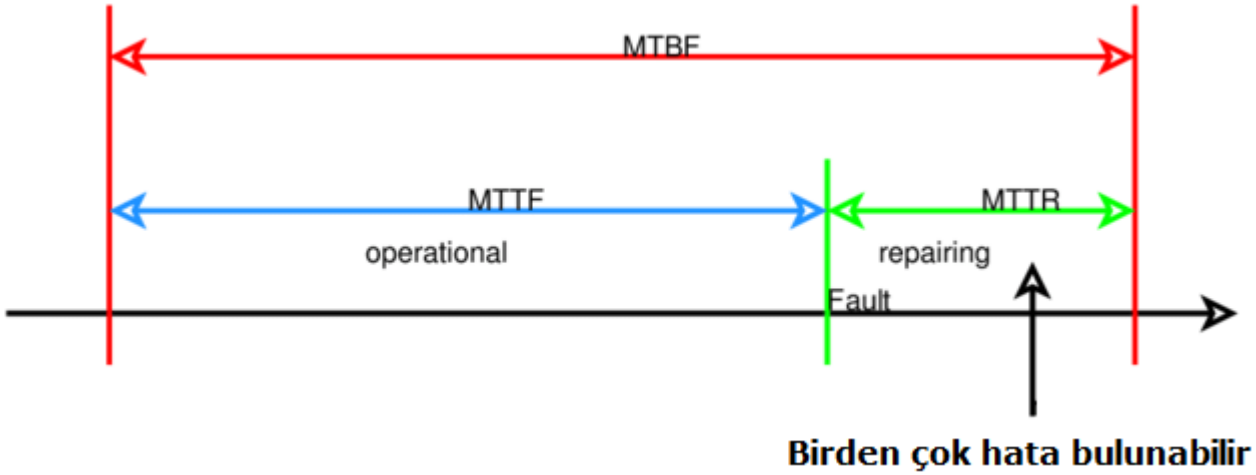
- Zaman yazılım güvenilirliğinde çok önemli bir noktada durmaktadır.
- Eğer bir yazılımda iki başarısızlık arasında çok az bir süre var ise düşük güvenilirlikli yazılım olarak isimlendirilir.
- Zamanın iki formu dikkate alınır.
  - Çalıştırma Zamanı
  - Takvim zamanı

Time Interval (Zaman Aralığı)

- MTTF: Mean Time To Failure – Başarısızlıkların Ortalama Zaman
- MTTR: Mean Time To Repair – Onarımların Ortalama Zamanı
- MTBF: Mean Time Between Failures (= MTTF + MTTR) Başarısızlıklar Arası Ortalama Zaman



Time Interval (Zaman Aralığı)



Yazılım Güvenirliğine İki Farklı Yaklaşım

- İlk Yaklaşım
  - Yazılım güvenirliği, belli bir zaman aralığında belli çerçevede başarısızlık durumuna **düşmeme** olasılığıdır.
  - Anahtar Kelimeler
    - Başarısızlık durumuna düşmeyen işlem olasılığı
    - Bu işlemin başarı ile çalışma süresi
    - Bu işlem için verilen çerçeve
  - Örnek: Bir marketteki bir kasa yazılımının 8 saat süre ile çökmeden çalışma olasılığı 0.97'dir.
    - 0.97
    - 8 saat
    - Bir Market

Yazılım Güvenirliğine İki Farklı Yaklaşım

- İkinci Yaklaşım
  - Yazılım güvenirliği, belli çerçevede başarısızlık yoğunluğu ile ölçülür.
  - Örnek: Hastane otomasyonu ayda 3 kez çöktü.

- Karşılaştırma
  - Birinci yaklaşım MTTF'ye önem verir.
  - İkinci yaklaşım başarısızlık sayısına

Güvenilirliği Ölçmek

- İki kabul görmüş yol bulunmaktadır.
  - Periyodik aralıklarla başarısızlıkları (failure) say
    - Kümülatif Başarısızlık Sayısı - **$\mu(\tau)$**
  - Başarısızlık Yoğunluğu
    - Birim zamandaki başarısızlık sayısı - **$\lambda(\tau)$**

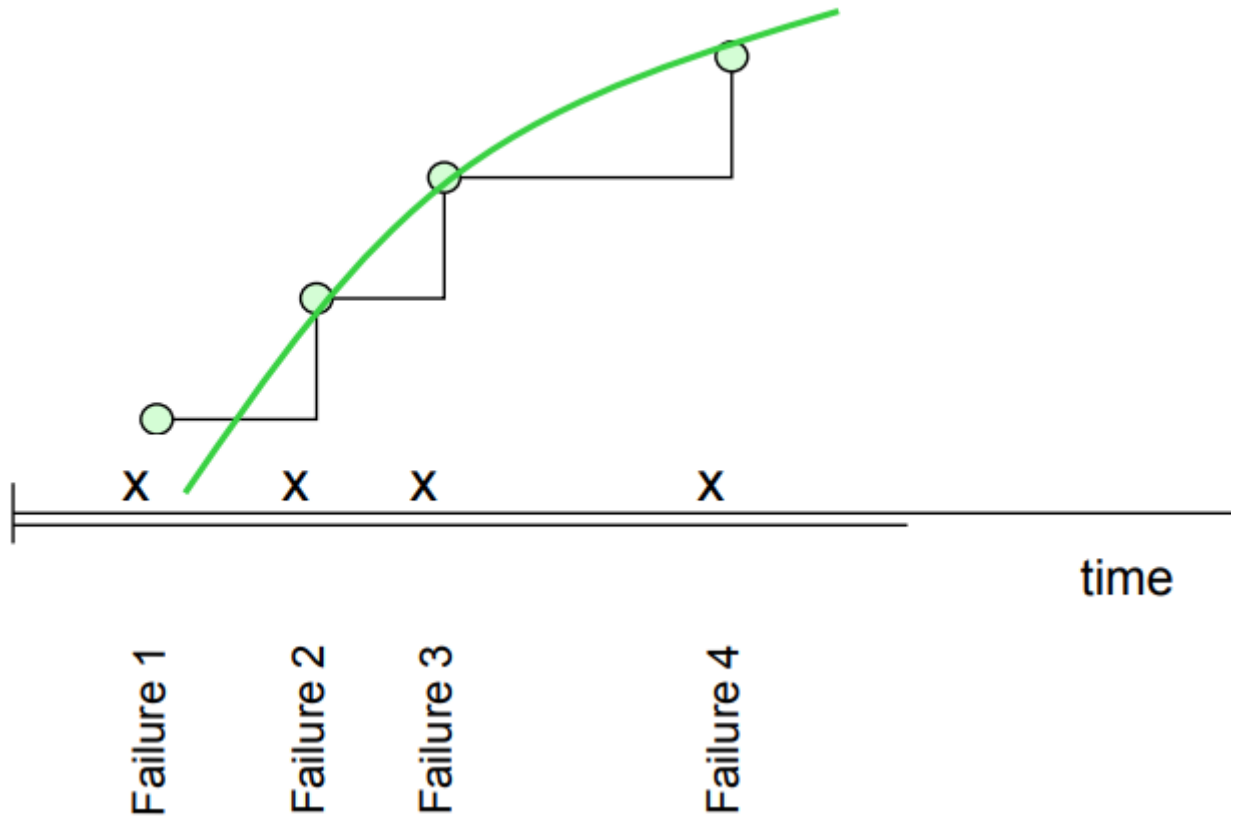
$\mu(\tau)$

Sistemin başlatılmasından yürütülme süresince toplam başarısızlık sayısı

$\lambda(\tau)$

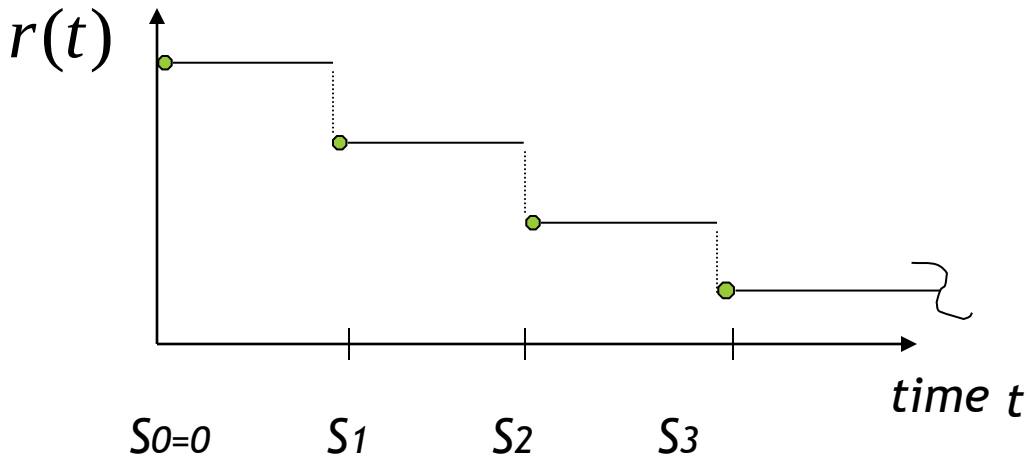
Sistemin başlangıçtan itibaren yürütülmesinden τ Zaman birim sonrası birim zamandaki başarısızlık sayısı

Kümülatif Başarısızlık Sayısı

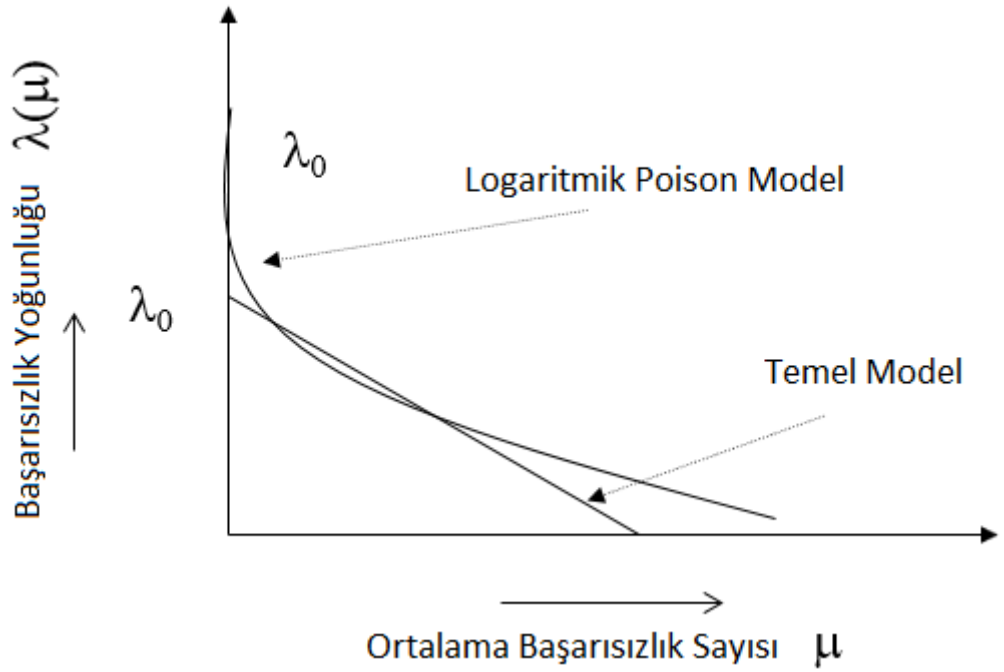


Güvenirliği Ölçmek

- $\lambda(\tau) = d\mu(\tau)/d\tau$



Güvenirliği Ölçmek



Hangi Model Kullanılmalıdır?

- Yazılımın düzenli işlem profili varsa
  - TEMEL MODEL
- Yazılımın düzensiz işlem profili varsa (Kompleks modeller)
  - LOGARITMIK POISON MODEL

Yazılım Güvenirliğinin Uygulamaları

- Yazılım Mühendisliği Teknolojilerinin Karşılaştırılması
  - Bir teknolojiye uyum maliyeti nedir?
  - İlgili teknolojiden maliyet ve kalite açısından geri dönüş nedir?
- Sistem Test Sürecinin Ölçülmesi
  - Test nasıl gerçekleştirildi?
  - Başarısızlık yoğunluğu ölçümü bize sistemin mevcut kalitesi hakkında bilgi verir.
  - Yüksek yoğunluk daha fazla test yapılması anlamına gelir.
- Sistemin Çalışır Durumda Kontrol Edilmesi
  - Bakım için bir yazılımdaki düzenleme oranı güvenirliği etkiler.
- Yazılım Geliştirme Süreçlerine Daha İyi Bakış
  - Kalitenin niceliği, geliştirme süreçlerine daha iyi bir bakış açısı sağlar.

Operasyonel Profiller (OP)

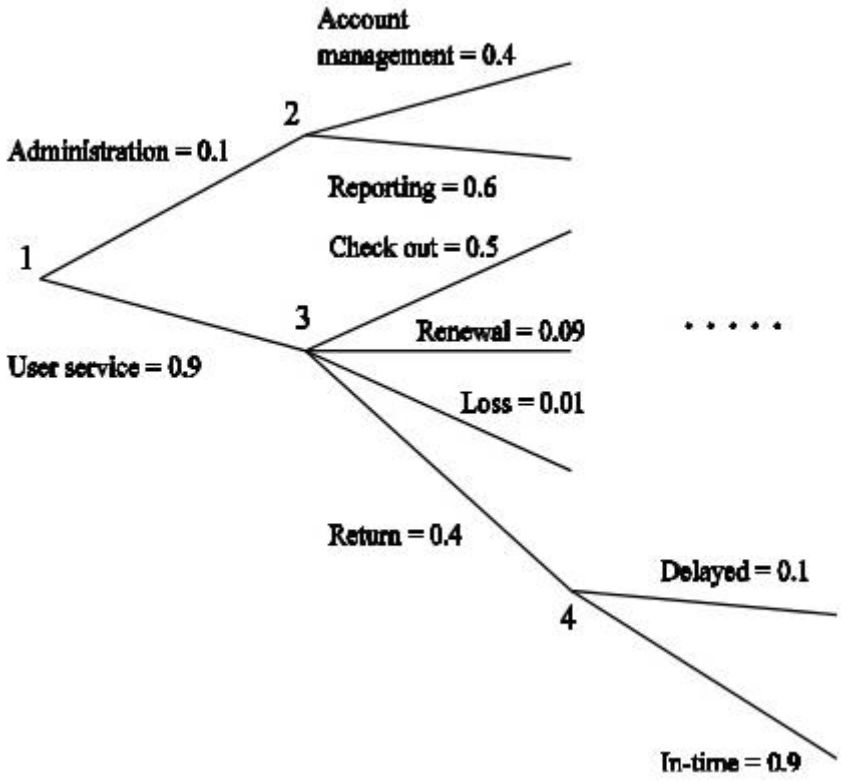
- AT&T Bell Laboratuvarlarında geliştirilmiştir.
- Bir OP gerçek kullanıcıların sistemi nasıl kullandıklarını açıklar.
- Bir OP aslında sayısal bir veridir.
- OP'yi temsil etmek için iki yöntem kullanılır.
  - Tablo olarak
  - Grafiksel

Tablo Olarak OP

Failure probability	R of comp (Signal from IR sensor)	R of comp (Activate obstacle Right)	R of comp (Activate obstacle Left)	R of comp (Activation finished)	R of comp (Set avoid active)	R of comp (Direction change finished)
0.1	0.835902	0.970920	0.977406	0.967953	0.967956	0.967743
0.08	0.865844	0.972942	0.978027	0.970594	0.970595	0.970374
0.06	0.896951	0.974942	0.978637	0.973201	0.973202	0.972974
0.04	0.929301	0.976923	0.979235	0.975778	0.975779	0.975542
0.02	0.962979	0.978884	0.979823	0.978326	0.978326	0.978082
0	0.998075	0.980828	0.980401	0.980846	0.980846	0.980594

doi:10.1371/journal.pone.0163346.t004

Grafiksel Olarak OP



Operasyonel Profillerin Kullanımları

- Bir sistemin güvenirliğinin doğru tahmin edilebilmesi için gerçek sahada kullanılacağı şekli ile test edilir. Bu da OP kullanarak sağlanabilir.
- Kullanıcı arayüzleri tasarlanırken bir yol gösterici olarak OP'ler kullanılır.
  - Sık kullanılan OP'ler kolay kullanım sağlamaktadırlar.
- Yazılımın erken sürümünü tasarlamak için OP kullanılır.
  - Sık kullanılan işlemleri içinde barındırır.
- Daha fazla kaynağın nereye yerleştirileceğini belirlemek için OP kullanılır.

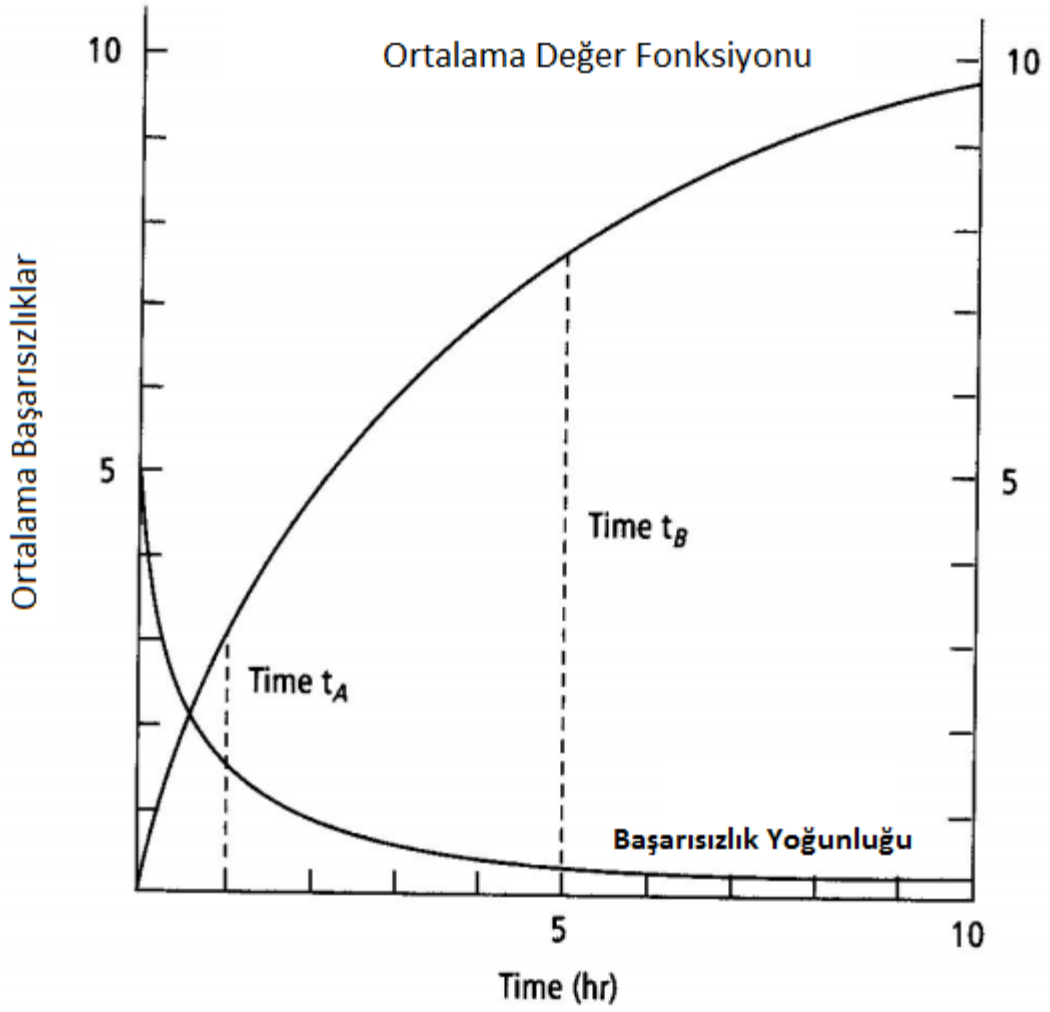
Güvenilirlik Modellerin Geliştirilmesinde Temel Varsayımlar

- Programdaki hatalar birbirinden bağımsızdır.
- Hatalar arasında yürütme süresi, komut yürütme süresine göre büyüktür.
- Potansiyel test alanı kullanım alanını kapsar.
- Her test için test verisi rastgele seçilir.
- Arızaya (fault) neden olan başarısızlık (failure) hemen giderilir. Ya da tekrar oluşması halinde başarısızlık sayısına dahil edilmez.

Güvenilirlik Modellerin Geliştirilmesi

- Tekrar oluşan aynı hata sayıya dahil edilmediği için başarısızlık yoğunluğu her hata düzeltildiğinde azalacaktır.
- Kümülatif başarısızlık sayısı arttıkça, başarısızlık yoğunluğu azalacaktır.
- Başarısızlık Yoğunluğunun azalmasında uygulanan iki işlem bulunur.
  - **Temel Model** : Gözlemlenen başarısızlığa neden olan ilişkili arıza sabitlendikten sonra başarısızlık yoğunluğunun azalması
  - **Logaritmik Model** : Gözlemlenen başarısızlığa neden olan ilişkili arızayı bir önceki azalmadaki değerden daha küçük bir değerde sabitledikten sonra başarısızlık yoğunluğunun azalması

Kümülatif Başarısızlık – Başarısızlık Yoğunluğu İlişkisi



Temel Model

Varsayım: $\lambda(\mu) = \lambda_0 (1 - \mu/v_0)$

$d\mu(\tau)/d\tau = \lambda_0 (1 - \mu(\tau)/v_0)$

$\mu(\tau) = \lambda_0 (1 - \mu/v_0)$

$\lambda(\tau) = \lambda_0.e^{-\lambda 0\, \tau/v0}$

λ_0: Sistem testinin başlangıcında görülen ilk arıza yoğunluğu

v_0: Sistem testinin başlangıcından itibaren belirsiz süre boyunca gözlemlemeyi umduğumuz toplam sistem hatası sayısı.

Logaritmik Model

Varsayım: $\lambda(\mu) = \lambda_0 e^{-\theta\mu}$

$d\mu(\tau)/d\tau = \lambda_0 e^{-\theta\mu(\tau)}$

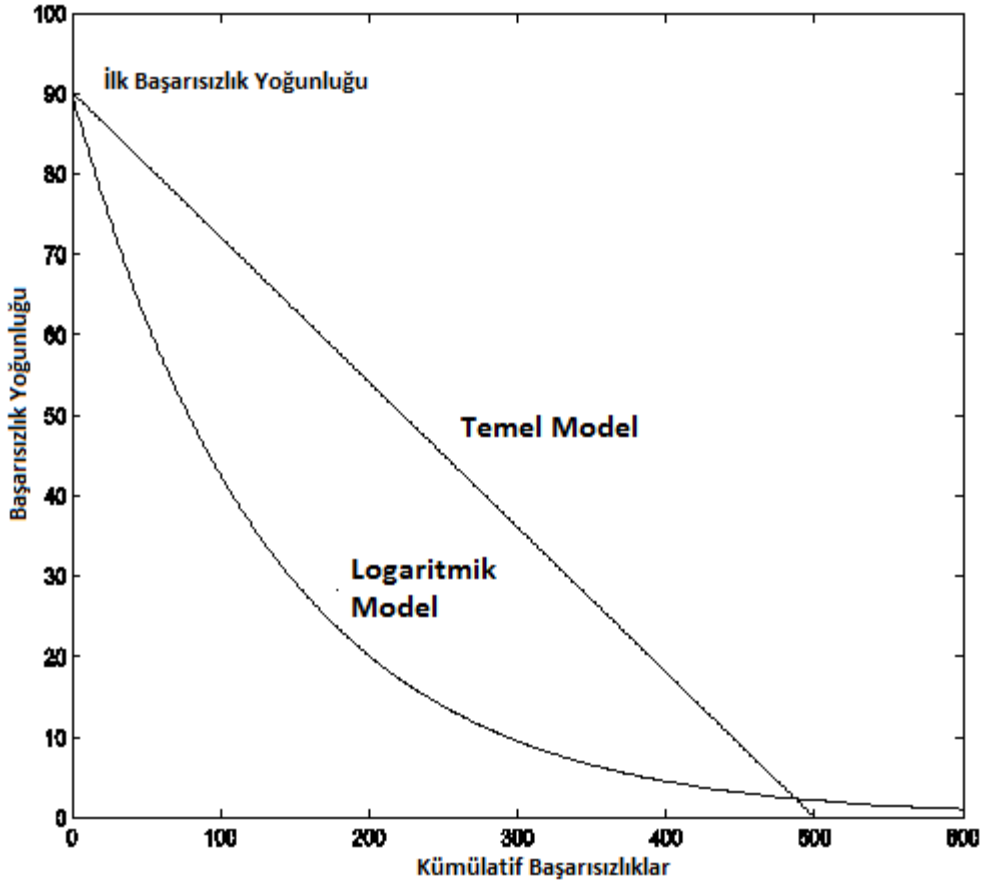
$\mu(\tau) = \ln(\lambda_0\theta\tau + 1)/\theta$

$\lambda(\tau) = \lambda_0/(\lambda_0\theta\tau + 1)$

λ_0: Sistem testinin başlangıcında görülen ilk arıza yoğunluğu

θ: Logaritmik modelde arıza yoğunluğunda doğrusal olmayan düşüşü temsil eden bir parametre.

Başarısızlık Yoğunluğunun Azalmasında İki Farklı İşlem



Başarısızlık Yoğunluğu (λ)
Çalışma Süresi (τ)
$\lambda_0 = 90$
$v_0 = 500$ Başarısızlık
$\theta = 0.0075$

Örnek

- Bir yazılım sistem testine tabi tutuluyor.
  - İlk Başarısızlık Yoğunluğu: 25 Başarısızlık / CPU saati
  - Şimdiki Başarısızlık Yoğunluğu : 5 Başarısızlık / CPU saati
  - Testin geçebilmesi için güvenirlilik seviyesine ulaşması için
    - 0.001 Başarısızlık / CPU saati
  - Tecrübelerden Bilinmeyen bir süre zarfında toplamda 1200 başarısızlığa ulaşılacağı öngörülüyor.
  - Yazılımın bu sistem testini geçebilmesi için gerekli olan saati hesaplayınız?

Cevap

- Soruda bilinmeyen bir süre zarfında (sonsuz süre) toplamda 1200 başarısızlığa ulaşılacağı öngörülüyor dediği için Temel model'e girmektedir.
- λ_c şimdiki başarısızlık yoğunluğu
- λ_r serbest bırakmadaki başarısızlık yoğunluğu.
- τ_c Bilinmeyen süre.
- λ_r , τ_r saat sonra erişildiği düşünülürse

$$\lambda_c = \lambda_0.e^{-\lambda 0\, \tau c/v0}$$
$$\lambda_r = \lambda_0.e^{-\lambda 0\, \tau r/v0}$$
$$\lambda_c / \lambda_r = (\lambda_0.e^{-\lambda 0\, \tau c/v0})/(\lambda_0.e^{-\lambda 0\, \tau r/v0}) = e^{(\tau r - \tau c)\, \lambda 0/v0}$$

$$\ln(\lambda_c / \lambda_r) = (\tau_r - \tau_c)\, \lambda_0/v_0$$

$$(\tau_r - \tau_c) = (v_0/ \lambda_0)\ln(\lambda_c / \lambda_r) = (1200/25)\ln(5/0.001) = 408.825 \text{ saat}$$

- Sistemin 0.001 Başarısızlık / CPU saati güvenirlilik seviyesine ulaşması için 408.825 saat gerekli

Örnek - 2

- İlk başarısızlık yoğunluğu: 25 başarısızlık / CPU Saati
- Düşüş parametresi : 0,025 / başarısızlık
- Bu süre zarfında 125 adet başarısızlık kaydedildi.
- Şuan ki başarısızlık yoğunluğu?
- 110 CPU saat sonrası Toplam başarısızlık sayısı?

Cevap

- Logaritmik modele göre
- λ_0= 25 başarısızlık / CPU saati
- $\mu(\tau)$ = 125 başarısızlık
- θ = 0,025 / başarısızlık
- Şuan ki başarısızlık yoğunluğu :
- $\lambda(\mu) = \lambda_0 / e^{-\theta\mu(\tau)}$
- $25 \times e^{-0,025 \times 125}$

=1,1 başarısızlık / CPU saati

Cevap devam…

- 110 CPU saat sonrası Toplam başarısızlık sayısı?

$\mu(\tau) = \ln(\lambda_0 \theta \tau + 1)/\theta$

$\mu(\tau) = \ln(25x0{,}025x110 + 1)/0{,}025$

$\mu(\tau) \approx$ 170 başarısızlık

Referanslar

- Naik, Kshirasagar, and Priyadarshi Tripathy. *Software testing and quality assurance: theory and practice*. John Wiley & Sons, 2011.
- Ammann, Paul, and Jeff Offutt. *Introduction to software testing*. Cambridge University Press, 2016.
- Padmini, C. "Beginners Guide To Software Testing." (2004).
- Archer, Clark, and Michael Stinson. *Object-Oriented Software Measures*. No. CMU/SEI-95-TR-002. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 1995.
- Pandey, Ajeet Kumar, and Neeraj Kumar Goyal. *Early Software Reliability Prediction*. Springer, India, 2015.
- Koskela, L. (2013). *Effective unit testing: A guide for Java developers*. Manning.