Kapsama Kriterlerine

Giriş

Hafta 1'den Hatırlayalım

TÜM Hataları Bulmak Neden İmkansız veya:
Yazılım Testi Neden Zordur?

1. Önemsiz olmayan herhangi bir program için tüm girdileri yürütmek **zorludur (inatçı)**.
2. Yazılımın her girdide sonlandırılmasının sağlanması **karar verilemez**
3. Karşılık gelen girdilere göre doğru/yanlış çıktıları tanımak, en azından ilk etapta yazılımı oluşturmak kadar zordur - **oracle problemi**

Bir Test Problemi

To: hoca@erciyes.edu.tr
From:babannemiyediler@erciyes.edu.tr
Subject: Test ile ilgili problem - Lütfen yardım edin!!!

Merhaba Hocam

Bize testlerin nasıl yazılacağını anlattınız, ancak ilk derste bize tüm girdileri deneyemeyeceğimizi veya her şeyi test etmeye çalışamayacağımızı söylemiştiniz.
Gerçekte neyi test edeceğime nasıl karar vereceğim?

Saygılarımla,
Ersoy

Bir Test Çözümü

To: ersoy_babannemiyediler@erciyes.edu.tr
From:hoca@erciyes.edu.tr
Subject: Re: Test ile ilgili problem - Lütfen yardım edin!!!

Merhaba Ersoy,

Korkmaya gerek yok.

Bir sonraki derste neyi test edeceğinize ve neyi kaçırdığınıza karar vermenize yardımcı olacak **kapsam kriterlerini** tanıtacağım.

Saygılarımla,
Hoca

Kapsama Kriterleri

Kapsama kriteri, bir yazılım parçasının soyut bir temsilini alır ve onu **test edilebilir özelliklere böler**.

Her özellik, yazılımın **test paketi tarafından test edilmesi gereken** bir **test gereksiniminin** temelini oluşturur.

Test paketindeki bir test senaryosu test gereksinimini karşıladığında, test gereksiniminin **kapsandığını (cover)** söyleriz.

Test paketinin kapsadığı test gereksinimlerinin yüzdesine **kapsam düzeyi** (veya daha basitçe "**kapsam**") denir.

Kapsam kriterleri, teste yönelik böl ve yönet yaklaşımıdır.

Bazı Sorular

Kapsama Kriterleri

Kapsama kriteri, bir yazılım parçasının soyut bir temsilini alır ve onu **test edilebilir özelliklere böler**.

Her özellik, yazılımın **test paketi tarafından test edilmesi gereken** bir **test gereksiniminin** temelini oluşturur.

Test paketindeki bir test senaryosu test gereksinimini karşıladığında, test gereksiniminin **kapsandığını (cover)** söyleriz.

Test paketinin kapsadığı test gereksinimlerinin yüzdesine **kapsam düzeyi** (veya daha basitçe "**kapsam**") denir.

Kapsam kriterleri, teste yönelik böl ve yönet yaklaşımıdır.

"Soyut temsil" derken neyi kastediyoruz?

"Test edilebilir özellikler" derken neyi kastediyoruz?

İfade (Statement) Kapsamı

En basit temsillerden biri, bir yazılım parçasını oluşturan **program ifadeleridir (statement)**.

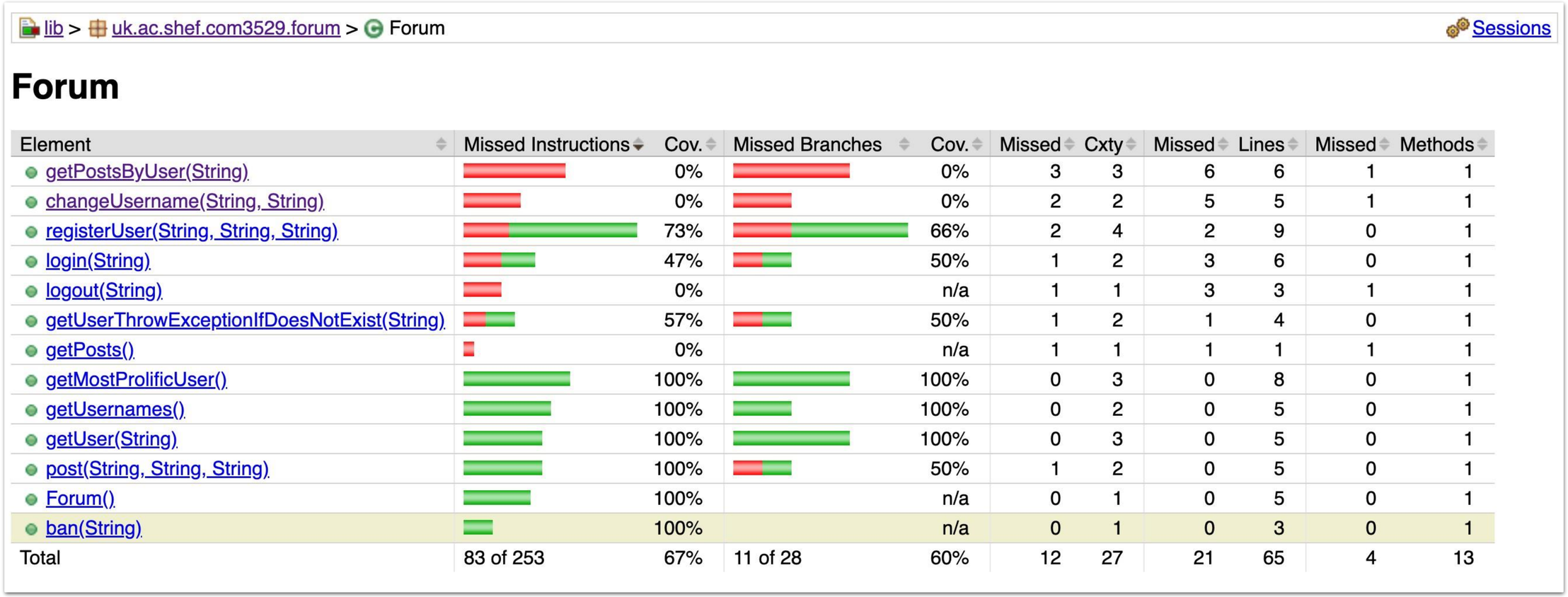
Her **özellik** bir program ifadesidir.

Her test gereksinimi, her satırın yürütülmesini (“kapsamasını”) içerir.

İfade kapsamı, test paketi tarafından yürütülen kod satırlarının yüzdesidir.

JaCoCo

Testi çalıştırdıktan sonra `./gradlew jacocoTestReport` komutunu çalıştırın daha sonra `code/lib/build/reports/jacoco` klasörüne gidin.

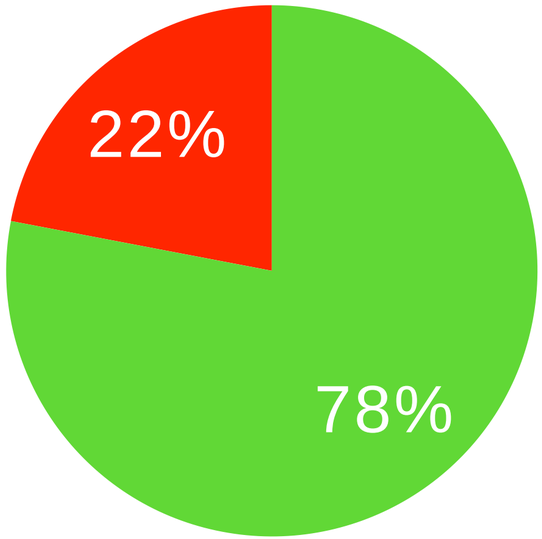


lib > uk.ac.shef.com3529.forum > Forum — Sessions

Forum

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
getPostsByUser(String)		0%		0%	3	3	6	6	1	1
changeUsername(String, String)		0%		0%	2	2	5	5	1	1
registerUser(String, String, String)		73%		66%	2	4	2	9	0	1
login(String)		47%		50%	1	2	3	6	0	1
logout(String)		0%		n/a	1	1	3	3	1	1
getUserThrowExceptionIfDoesNotExist(String)		57%		50%	1	2	1	4	0	1
getPosts()		0%		n/a	1	1	1	1	1	1
getMostProlificUser()		100%		100%	0	3	0	8	0	1
getUsernames()		100%		100%	0	2	0	5	0	1
getUser(String)		100%		100%	0	3	0	5	0	1
post(String, String, String)		100%		50%	1	2	0	5	0	1
Forum()		100%		n/a	0	1	0	5	0	1
ban(String)		100%		n/a	0	1	0	3	0	1
Total	83 of 253	67%	11 of 28	60%	12	27	21	65	4	13

Kapsama Seviyesi



JaCoCo, kapsamı Java ByteCode düzeyinde hesapladığı için ifadelerin aksine talimatları (**instructions**) ölçer

78% ifade yürütüldü (covered)

22% ifade yürütülmedi (uncovered)

Kapsam düzeyi (veya yalnızca "kapsam"), test paketi tarafından karşılanan test gereksinimlerinin yüzdesidir.

Burada %78 ifade yürütülür, dolayısıyla **ifade kapsamı %78 olur.**

JaCoCo Sınıf Kapsamı Raporu

```
1.  package uk.ac.shef.com3529.forum;
2.
3.  import java.time.Instant;
4.  import java.util.*;
5.
6.  public class Forum {
7.
8.      Set<User> users;
9.      Set<User> online;
10.     List<Post> posts;
11.
12.     public Forum() {
13.         users = new HashSet<>();
14.         online = new HashSet<>();
15.         posts = new LinkedList<>();
16.     }
17.
18.     User getUser(String username) {
19.         for (User user : users) {
20.             if (user.getUsername().equals(username)) {
21.                 return user;
22.             }
23.         }
24.         return null;
25.     }
26.
27.     User getUserThrowExceptionIfDoesNotExist(String username) {
28.         User user = getUser(username);
29.         if (user == null) {
30.             throw new UnknownUserException("Unknown user " + user);
31.         }
32.         return user;
33.     }
```

Yeşil = test(ler) tarafından **yürütülmüş**

Sarı = test(ler) tarafından `if` veya `for`/`while` ifadelerinin dallarından (***branch***) biri (true veya false) yürütülmüş

Kırmızı = test(ler) tarafından **yürütülmemiş**

Eleştiriler ve Diğer Kapsam Kriterleri

İfadeler, yazılımı temsil etmenin ve onu test için bölmenin (ve ardından "fethetmenin") çok basit bir yoludur.

İfade Kapsamı, **"neyin test edileceğine karar verme" yöntemi değil**, **daha ziyade bir test paketinin kapsamına girmeyenleri keşfetmenin ve daha fazla test senaryosunun gerekli olup olmadığına karar vermenin bir yoludur**.

Ders boyunca farklı (ve potansiyel olarak daha faydalı) kapsam kriterlerini inceleyeceğiz.

%100 Kapsama Sahip Olmak Programımızın Hatasız Olmasını Sağlar mı?

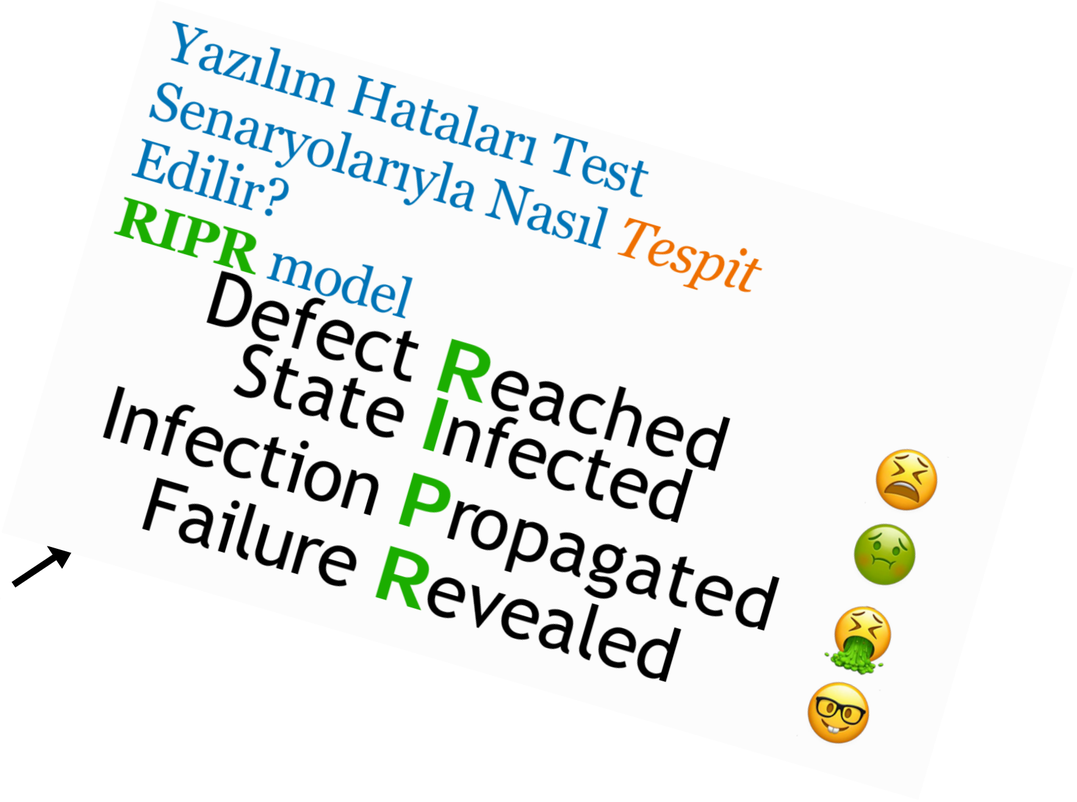
%100 Kapsama Sahip Olmak Programımızın Hatasız Olmasını Sağlar mı?

HAYIR! – ifade kapsamı gibi kapsam kriterleri, test gereksinimlerinin sistematik bir şekilde oluşturulmasına yönelik bir yöntemdir.

Örneğin, %100 ifade kapsamı kusurların çoğunu çalıştırabilir ancak:

- kusurların programın durumunu etkileyeceğini **garanti etmez**
- enfeksiyonların programın çıktısına hata olarak yayılacağını **garanti etmez**
- hataların test paketinin assertionları tarafından tespit edileceğini **garanti etmez**

RIPR modelini hatırlayın



Uygulanamaz Test Gereksinimleri

Ayrıca bazı test gereksinimleri **uygulanamayabilir (infeasible)**.

Yani bu kapsanmasının imkansız olduğu anlamına gelir.

Örneğin, yürütülmesi imkansız olan bir kod satırı, yani ölü kod (dead code).

```
if (false) {
  aMethodCallThatNeverHappens();
}
```

Bu program ifadesi hiçbir zaman yürütülemez! İfade kapsamı için uygulanamayan (infeasible) bir kapsam gereksinimi oluşturur.

Özet: Önemli Terminoloji

Kapsam Kriteri: Yazılımı test için bir dizi test gereksinimlerine bölme yöntemi.

Test Gereksinimi: Test paketinin bir kapsam kriterine göre yerine getirmek zorunda olduğu bir yazılım parçasının özelliği. Bir test gereksiniminin yazılım gereksinimiyle ve bir test senaryosu ile aynı olmadığını unutmayın. Bir test senaryosu birçok test gereksinimini karşılayabilir.

Uygulanamaz Test Gereksinimi: Yerine getirilmesi imkansız bir test gereksinimi.

Kapsama Düzeyi: Bir test paketi tarafından yürütülen (kapsanan) test gereksinimlerinin yüzdesi.

Saydam, Kara ve Gri Kutu Kapsamı Kriterleri

Kapsam kriterlerini kategorize etmenin bir yolu, test gereksinimlerinin koddan mı türetildiğini yoksa programlanma şekline herhangi bir atıfta bulunup bulunmadığını ayırt etmektir.

Saydam Kutu

Dahili kod yapısı hakkında tam bilgi. Bu nedenle sıklıkla “kod kapsamı” kriterleri olarak anılırlar.

Kara Kutu

Dahili kod yapısı hakkında bilgi yok. Yazılımın gereksinimlerine, tasarımlarına veya soyut modellerine dayalı kapsam kriterleri.

Gri Kutu

Dahili kod yapısı hakkında biraz bilgi. Kapsam kriterleri, "beyaz kutu" ve "kara kutu" olarak değerlendirilen çıktıların bir karışımını temel alır.

Farklı Kriterler, Farklı Kusurlar

Devreye alma kusuru

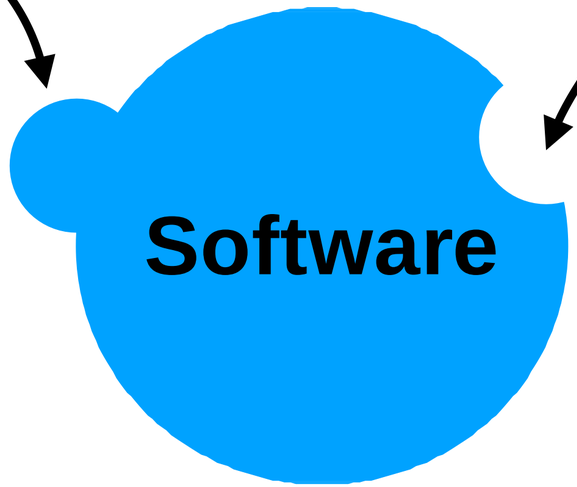
Kod yapması gerekenden fazlasını yapar

örneğin, bir dizinin sonunun ötesinde yinelenen ve arabellek taşmasına neden olan bir C programı

İhmal kusuru

Kod yapması gerekenden azını yapar

Örneğin gereksinimin uygulanmaması



Saydam Kutu Kapsam Kriterleri

Kodda "görebildiğiniz" gibi, **devreye alma kusurlarını tespit etmek için en uygunudur.**

ihmalden kaynaklanan kusurları tespit etmeye daha az uygundur.

Kara Kutu Kapsam Kriterleri

Yazılımın dahili işleyişi hakkında bilgi olmadığından **devreye alma kusurlarını tespit etmeye daha az uygundur.**

Gereksinimlere ve soyut yazılım modellerine dayalı olarak ihmal hatalarını tespit etmek için en uygunudur.