



YMT 412-Yazılım Kalite Güvencesi ve Testi

Yazılım Metrikleri

Prof. Dr. Resul DAŞ
Fırat Üniversitesi Yazılım Mühendisliği Bölümü

Bölüm-9

İçindekiler

	Yazılım Ürün Metrikleri.....	17
	Metrik Tanımları.....	21
	Geleneksel Metrikler.....	28
	Nesne Tabanlı Metrikler.....	34

4. Yazılım Ürün Metrikleri

- Kelime anlamı olarak metrik, ölçülebilen değerlerdir. Yazılımların, test edilebilmeleri ve sonrasında müşteriye teslim edilebilmesi için belli bir olgunluğa gelmesi gereklidir. Yazılımın ve testin belli bir olgunluğa geldiğini “Metrik” denilen değerler ile ölçümleriz.

4. Yazılım Ürün Metrikleri

➤ Ürün metrikleri:

- Proje durumunun izlenmesi ve gözlenmesinde kullanılır.
- Bir projenin özkaynak gereksinimi için tahminlerde bulunabilme olanağı verirler.
- Personelin başarımını değerlendirme ve sorgulayabilme olanağı sağlarlar.
- Örgüt yapılarını, bireysel çalışma yöntemlerini değerlendirebilme olanağı verirler.
- Yazılım geliştirme yardımcı araçlarının karşılaştırılmasında ya da onların tasarımında bir temel oluştururlar.
- Yanlış ya da ters giden şeylerin bulunmasına yardımcı olurlar.

4. Yazılım Ürün Metrikleri

➤ **Andaç metrikler:** Bir andaç(token), kaynak kodu oluşturan basit bir birim olarak değerlendirilebilir. Andaç olarak if, for, while, switch, class, procedure gibi programlama diline bağlı bir anahtar sözcük seçilebileceği gibi özel bir tanımlayıcı ya da bir değişken ismi seçilebilir. Andaç metrikler bir sistemin tümünün, bir parçasının ya da bir yazılım biriminin kaynak kodu içinde bulunan bu tür andaçların sayılmasıyla hesaplanır. Sayım sonunda istatistiksel bilgiler elde edilerek bir karşılaştırma yapılır.

4. Yazılım Ürün Metrikleri

- **Denetim Akış Metrikleri:** Bu metrikle bir yazılım sistemi içinde yer alan her yazılım birimi incelenerek denetim akışı belirlenir. Çözümleme yapılan birimin denetim yapısının karmaşıklığı grafiksel gösterimle yapılır. Giren ve çıkan bilgi akışının, iletilerin sayısı bir metrik oluşturur.
- **Bileşik Metrikler:** Andaç ve denetim akış metriklerinin birleştirilerek kullanılmasıyla oluşturulur.
- **Sistem Metrikleri:** Diğer metrikler yazılım kodunun çeşitli özellikleriyle ilgilenirken sistem metrikleri bir sistemin daha çok tasarım niteliğiyle ilgili süreç, boyut, zamanlama, ürün niteliği, bakılabilirlik, üretkenlik gibi büyük ölçekli özellikleri üzerinde yoğunlaşır. Bu metrikler bir projenin erken evrelerinde başladığı için geleceğini tahmin etmede büyük yarar sağlarlar.

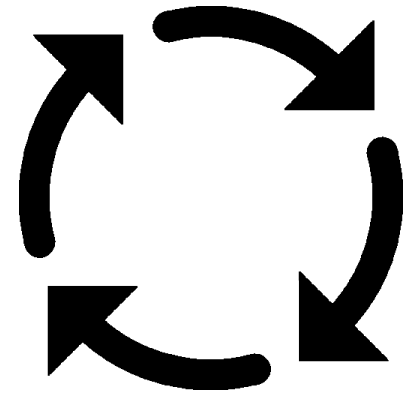
5. Metrik Tanımları

➤ Toplanıp değerlendirilecek her metriğin belirli bir amacının olması, belirli sorulara yanıt bulabilmesi ve değerlendirilmesi gereklidir. Andaç, denetim akış ve bileşik metrikler küçük projelerde uygulama alanı bulurlar. Büyük projelerde ise sistem metrikleri daha büyük önem taşır. Bu nedenle yaygın olarak kullanılan sistem metriklerinden bazılarının tanımını göreceğiz.

- Süreç metrikleri
- Boyut metrikleri
- Zamanlama metrikleri
- Maliyet ve kaynak metrikleri
- Ürün nitelik metrikleri
- Bakım ve okunabilirlik metrikleri
- Üretkenlik metrikleri

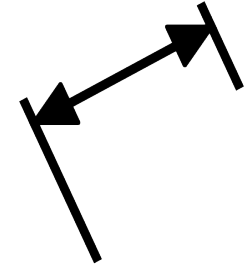
5. Metrik Tanımları

➤ **Süreç Metrikleri:** Yazılım geliştirme sürecinin niteliğini izlemek üzere, kullanılan tekniklerin, araçların insanların, örgütün ve altyapısının özelliklerinden oluşan metrikler kullanılır. Bu şekilde, bir sürecin doğru uygulanıp uygulanmadığı, bir yazılım aracının etkinliği ve üretkenliği, kodlayıcıların yetenekleri, örgütün içsel ve dışsal iletişim yetenekleri, altyapının ve tesislerin yeterlilikleri değerlendirilebilir.

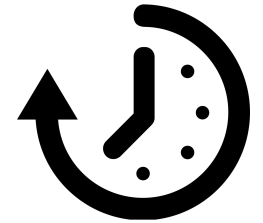


5. Metrik Tanımları

➤ **Boyut Metrikleri:** Boyut metriklerinin hedefi projenin büyüklüğüne ilişkin durumun önceden yapılan planla uyuşmakta olduğunun gözlemlenmesidir. Planlanan toplam ister sayısı ile değerlendirmenin yapıldığı anda karşılanmış olan ister sayısı karşılaştırılarak isterlerin durumu gözlemlenir.



➤ **Zamanlama Metrikleri:** Zamanlama metriklerinin toplanma amacı projenin takvime uygun yürütülmekte olduğunun gözlemlenmesine yardımcı olmaktır. Kullanılan en önemli ölçüt zamandır. Projenin tahmin edilen toplam süresi ile o ana kadar harcanan süre projenin durumu hakkında bilgi veren önemli bir göstergedir.



5. Metrik Tanımları

- **Maliyet Ve Kaynak Metrikleri:** Projenin belirli bir aşamasında, önceden kestirilen kişi- zaman cinsinden emek ile o tarihe kadar gerçekleşen emek karşılaştırılarak bir metrik elde edilebilir. Benzer şekilde harcanan para da karşılaştırılabilir.



5. Metrik Tanımları

➤ **Ürün Nitelik Metrikleri:** Amacı ürünün doğru ve kusursuz olarak üretildiğinden emin olmaktır. Bu maksatla, karşılaşılan yazılım kusurlarının doğaları, türleri, oluşma nedenleri ve ne zaman oluştukları kayıt altına alınır. Kusurların önem dereceleri, oluşma yoğunlukları ve bunların yazılımın büyüklüğüne oranları her bir aşama için belirlenir. Kusur belirleme sürecinin de etkinliği ve niteliği ürün piyasaya sürülmeden önce ortaya çıkarılan kusur sayısı ile ölçülür.

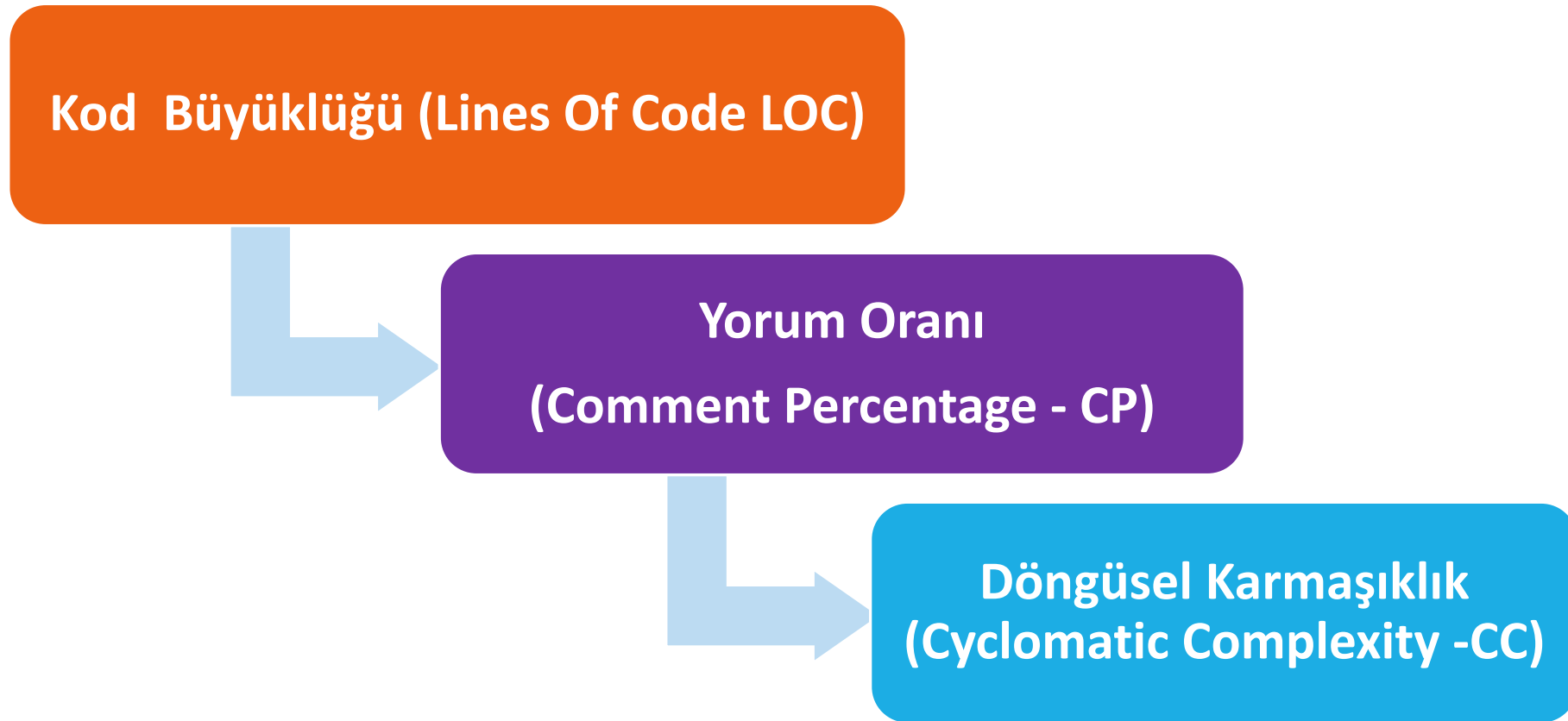
5. Metrik Tanımları

- **Bakım Ve Okunabilirlik Metrikleri:** Yapılan çeşitli araştırmalara göre yazılım bakım ve okunabilirliğini artırabilmek üzere kodlama sırasında dikkate alınabilecek bazı sayısal değerler verilebilir. Bunlardan bazıları şunlardır:
 - *Yordamların içinde yer alan toplam satır sayısı 60'ı, çalışır deyimlerin sayısı da 50'yi geçmemelidir.*
 - *Açıklama satırları yordam satır sayısının en az yarısı kadar olmalıdır.*
 - *Bir değişkene yapılan atama ile o değişkenin bir yordam içinde kullanılması arasındaki satır sayısı 10'dan fazla olmamalıdır.*
- Bu tarz özellikler dikkate alınarak metrik oluşturulur. Buna göre geliştirilen yazılımların karşılaştırılması yapılır.

5. Metrik Tanımları

- **Üretkenlik Metrikleri:** Her proje için önemli bir etken olan üretkenlik durumu, tüm proje boyunca dikkatle takip edilmeli, bu amaçla her aşamadaki genel üretkenliği ölçebilecek metrikler tanımlanmalıdır. Bunlardan bazıları şunlardır:
- Kazanılmış değer hesaplamaları
- Bir aşamada ortaya çıkan hataların harcanan kişi-zaman cinsinden emeğe oranı
- Bir aşamada elde edilen boyut metriklerinin emeğe oranı

6. Geleneksel Metrikler



6. Geleneksel Metrikler

Kod Büyüklüğü (Lines of Code-LOC)

Geleneksel olarak yazılımın boyutu satır sayısı ile ölçülür. Bu ölçümün çeşitli şekilleri vardır.

- **Satır Sayısı (Lines of Code-LOC):** Programın tüm satırlarının sayılmasıdır.
- **Yorum Ve Boşluk İçermeyen Satır Sayısı (Non-comment Non-blank-NCNB):** Programın yorum satırları ve boş satırlardan arındırılmış halidir.
- **Çalıştırılabilir Yordam Sayısı (Executable Statements-EXEC) :** Program içinde yer alan yordam sayısıdır.

6. Geleneksel Metrikler

Örneğin aşağıdaki kod :

```
IF X = 3
```

```
// Comment Line
```

```
Then
```

```
Y = 10
```

4 LOC , 3 NBNC ve 1 EXEC
olarak sayılır.

➤ Satır büyüklüğü pek çok dile uyarlanabildiği için oldukça kullanışlı ve ölçümü beklenen bir metriktir. Ancak satır büyüklüğü üzerine yorum yapmak için dilin karmaşıklığı, projenin karmaşıklığı gibi faktörler de göz önünde bulundurulmalıdır.

➤ Satır büyüklüğünün küçük olması yazılım için hedef olmalıdır.

6. Geleneksel Metrikler

Yorum Oranı (Comment Percentage - CP)

- Yorum oranı, program için hazırlanmış yorum satırlarının, toplam programın yorum ve boşluk içermeyen satır sayısına bölümü ile bulunur. Yüksek yorum oranı programların anlaşılabilirliğini arttıran ve bakımını kolaylaştıran bir faktördür.
- Yorum oranının %20 ila %30 arası olması tavsiye edilen bir durumdur.

6. Geleneksel Metrikler

➤ Döngüsel Karmaşıklık (Cyclomatic Complexity -CC)

- Thomas J. McCabe tarafından ortaya konan bu metrik bir metodun içindeki algoritmanın karmaşıklığını ölçmek için kullanılır. Aynı zamanda bir metodun test edilmesi için gerekli test durumu sayısını da verir.

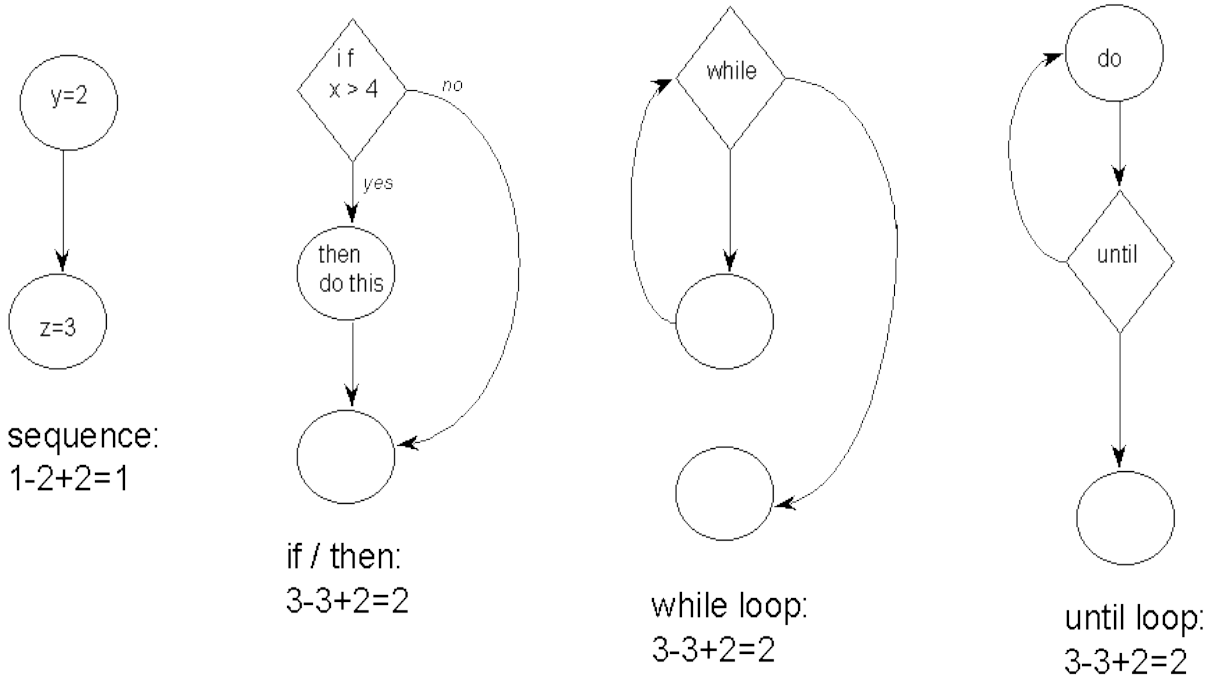
$$CC = \text{yollar} - \text{düğümler} + 2$$

- şeklinde hesaplanır. Örneğin bir IF cümlesi iki seçeneğe sahiptir. Eğer şart doğru ise birinci yol test edilmiş olur , hatalı ise ikinci yol test edilir.

6. Geleneksel Metrikler

Cyclomatic Complexity

Number of Independent Test Paths $\Rightarrow$ edges - nodes + 2



➤ Metotların az karmaşıklığa sahip olması tercih edilir. Bu kodun anlaşılabilirliğinin ve test edilmesinin kolay olduğunu gösterir. Bir metot için karmaşıklığın 10'u aşmaması tercih edilmelidir. Ancak 20'ye kadar kabul edilebilir. Bunu aşan değerler anlaşılması oldukça güç kodları gösterir.

7. Nesne Yönelimli Metrikler

Chidamber ve Kemerer Metrikleri

1. Sınıf Başına Ağırlıklı Method (Weighted Methods per Class -WMC)
2. Kalıtım Ağacının Derinliği (Depth of Inheritance Tree - DIT)
3. Alt Sınıf Sayısı (Number of Children - NOC)
4. Nesneler Arası Eşleme (Coupling Between Object Classes - CBO)
5. Sınıf Yanıt Sayısı (Response for a Class - RFC)
6. Uyumsuzluk (Lack of Cohesion - LCOM)

7. Nesne Yönelimli Metrikler

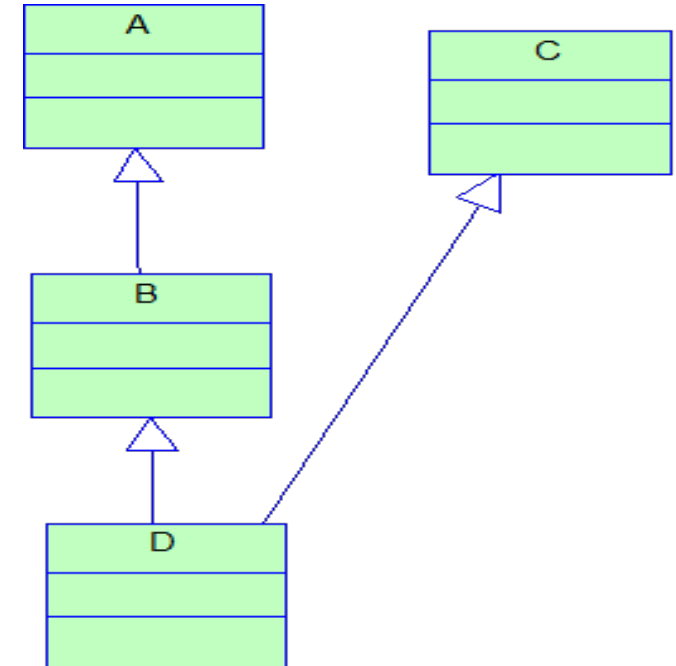
1. Sınıf Başına Ağırlıklı Method (Weighted Methods per Class -WMC)

➤ **Döngüsel karmaşıklık değerleri toplamının sınıf sayısına bölümü ile bulunur.** WMC değerinin 100'ün altında olması tercih edilmelidir. WMC sınıf karmaşıklığı hakkında fikir veren en önemli donelerden biridir. Bu metriğin faydaları sınıf başına ortalama metot sayısı gibi değerlendirilebilir. WMC değerinin 100'ün altında olması kabul edilebilir bir değerdir. WMC değeri NOM'dan farklı olarak sınıfın karmaşıklığı hakkında daha net bir fikir verir.

7. Nesne Yönelimli Metrikler

2. KALITIM AĞACININ DERİNLİĞİ (DEPTH OF INHERITANCE TREE - DIT)

➤ DIT metriği sınıfın ebeveyn sınıflarının sayısını gösterir. Eğer çoklu kalıtım durumu söz konusu ise bu durumda hiyerarşideki en uzun yol kabul edilir. Örneğin yandaki şekilde yer alan D sınıfı için DIT değeri 2 olarak kabul edilir. DIT bize kısaca kaç tane ana sınıfın potansiyel olarak sınıfımızı etkileyebileceğini gösterir.



7. Nesne Yönelimli Metrikler

- Sınıf hiyerarşisinin derinliği arttıkça daha fazla metot kalıtım alınır ve sınıfın davranışlarını öngörmek; anlamak zorlaşır.
- Derin hiyerarşiler daha fazla metot ve sınıfı etkilendiğinden, dizaynı karmaşıktır. Ancak hiyerarşi derinleştikçe kalıtım alınan metotların yeniden kullanım potansiyeli artar.
- **DIT için önerilen rakam genellikle 5'in altında olmasıdır.** 5'in üzerindeki derinlikler oldukça karmaşık yapılar doğurabilir. DIT'in 0 olması sınıfın kök olduğunu gösterir. DIT'in ortalama 2-3 arası bir değerde olması yeniden kullanımın iyi seviyede olduğunu gösterir. 2'den küçük derinlikler ise yeniden kullanımın zayıf olduğu alanları işaret eder.

7. Nesne Yönelimli Metrikler

3. Alt Sınıf Sayısı (Number of Children - NOC)

- NOC metriği sınıftan türemiş alt sınıflarının sayısını verir.
- Alt sınıf sayısı çoğaldıkça kalıtım özelliğine bağlı olarak yeniden kullanımın arttığı anlaşılr.
- Alt sınıf sayısının çokluğu sınıfın hatalı soyutlama yapıldığını, belki de hatalı bir hiyerarşi kurulduğunu gösterebilir.
- NOC sınıfın nüfus alanı hakkında bir fikir verir. **NOC metriği yüksek sınıflar gözden geçirme, test gibi süreçlerin daha dikkatli ve uzun tutulması gereken yerlerdir.**

7. Nesne Yönelimli Metrikler

4. Nesne sınıfları arasındaki bağımlılık (Coupling Between Object Classes - CBO)

- Bir sınıf içindeki özellik ya da metotların diğer sınıfta kullanılması ve sınıflar arasında kalıtımın olmaması durumunda iki sınıf arasında bağımlılıktan bahsedilebilir. **CBO; verimliliği ve yeniden kullanılabilirliği ölçmede kullanılır.**

7. Nesne Yönelimli Metrikler

5. Sınıfın tetiklediği metot sayısı (Response for a class - RFC)

- Bir sınıftan bir nesnenin metotları çağırılması durumunda, bu nesnenin tetikleyebileceği tüm metotların sayısı RFC değerini verir. Yani, bir sınıfta yazılan ve çağırılan toplam metot sayısıdır. **Bu metrik; sınıf seviye tasarım metriklerinden olup; anlaşılabilirliği, dayanıklılığı, karmaşıklığı ve test edilebilirliği ölçmede kullanılır.**

7. Nesne Yönelimli Metrikler

6. Metotlardaki uyum eksikliği (Lack of cohesion in methods - LCOM)

➤ LCOM, n adet kümenin kesişiminden oluşan kümelerdeki uyumsuzlukların sayısıdır ve metotlardaki benzerlik derecesini ölçer. Metotlardaki uyum eksikliği; bir sınıfın, iki veya daha fazla alt sınıfa ayrıldığını gösterir ve karmaşıklığı arttırır. Yapılan bir çalışmada, LCOM ölçütünün uyum özelliğini çok da iyi ayırt edemediği ispatlanmıştır. **LCOM metriği test ediciye; verimlilik ve yeniden kullanılabilirlik derecesi hakkında bilgi verir.**

Sorularınız



Çalışma Soruları

1. Yazılım geliştirmede ürün süreç ve niteliği ne demektir?
2. Yazılım ürün metrikleri ne için kullanılır?
3. Bir internet sitesi oluşturmada kullanılabilecek metrikler tanımlayınız.
4. Nesne tabanlı uygulama geliştirmede kullanılan metrikleri açıklayınız.
5. Gözden geçirmelere neden ihtiyaç duyulur? Test yapılan bir uygulamada gözden geçirme yapmak gerekli midir?
6. Geleneksel metrik türleri nelerdir? Adlarını yazıp açıklayınız.
7. Eşdüzey gözden geçirme nedir? Kimler arasında yapılır?

Kaynaklar

- [1] Software Engineering A Practitioner's Approach (7th Edition), Roger Pressman, 2013
- [2] Yazılım Mühendisliği (2. Baskı), M. Erhan Sarıdoğan, 2008
- [3] http://www.tutorialspoint.com/software_testing/software_testing_quick_guide.htm
- [4] <http://www.testrisk.com/2012/08/test-sozlugu.html>
- [5] https://en.wikipedia.org/wiki/Security_testing
- [6] <http://www.slideshare.net/pbaskarmca/security-testing-4338585>
- [7] <http://blog.milliyet.com.tr/yazilim-kullanilabilirlik-testleri/Blog/?BlogNo=463001>