Nesneye Dayalı Yazılım Metrikleri ve Yazılım Kalitesi

Ural ERDEMİR, Umut TEKİN,
Feza BUZLUCA

İÇERİK

- Yazılımda Kalite Kavramları
- Yazılımın İç Özellikleri
- Metrik Kümeleri
- Düşük Kalite Göstergeleri
- Tasarım Kalıpları
- Yardımcı Araçlar
- Değerlendirme ve Sonuç

Yazılım Projeleri

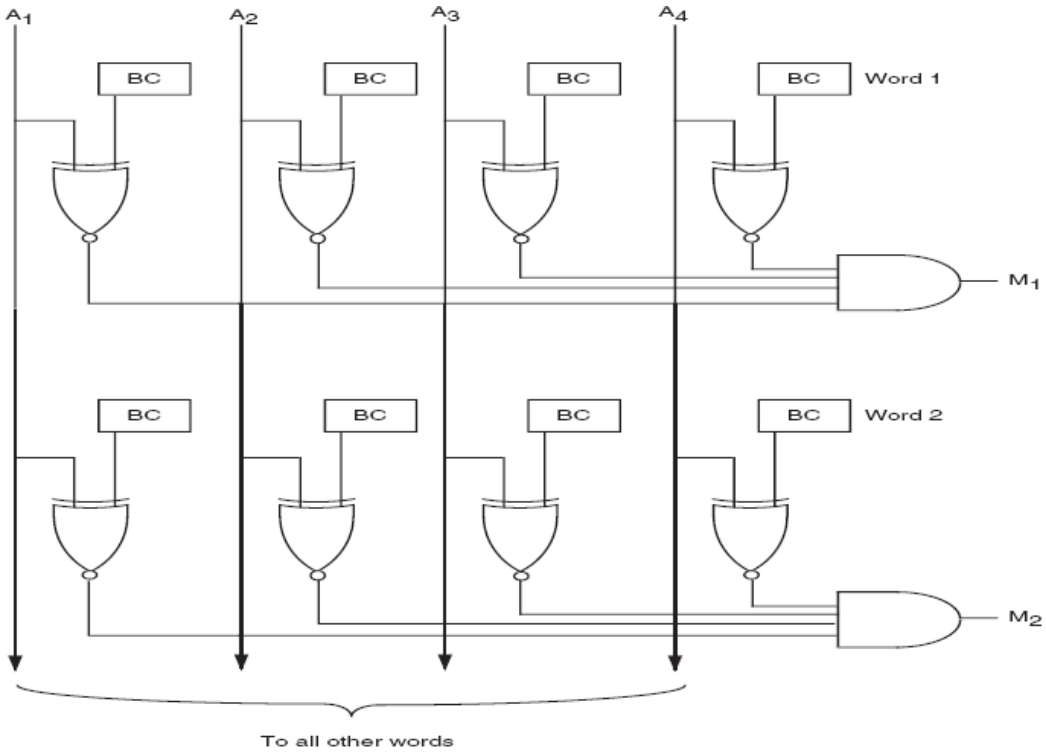
- Amerikan ordusunun yazılım projeleri üzerine yaptığı bir araştırmaya göre yapılan yazılım projelerinin;
  - %47'si kullanılmamakta
  - %29'u müşteri tarafından kabul edilmemekte
  - %19'u başladıktan sonra iptal edilmekte ya da büyük ölçüde değiştirilerek yeniden başlatılmakta
  - %3'ü kimi değişikliklerden sonra kullanılmakta
  - %2'sinin ancak teslim alındığı gibi kullanılmakta olduğu görülmüştür.

Hata - Maliyet

- 2002 yılında NIST tarafından yapılan bir araştırma sonucunda yazılım hatalarının Amerikan ekonomisine yıllık maliyeti **59 Milyar $** olarak tahmin edilmiştir ki o yıllarda satılan yazılımların toplam değeri **180 Milyar $** civarındaydı.

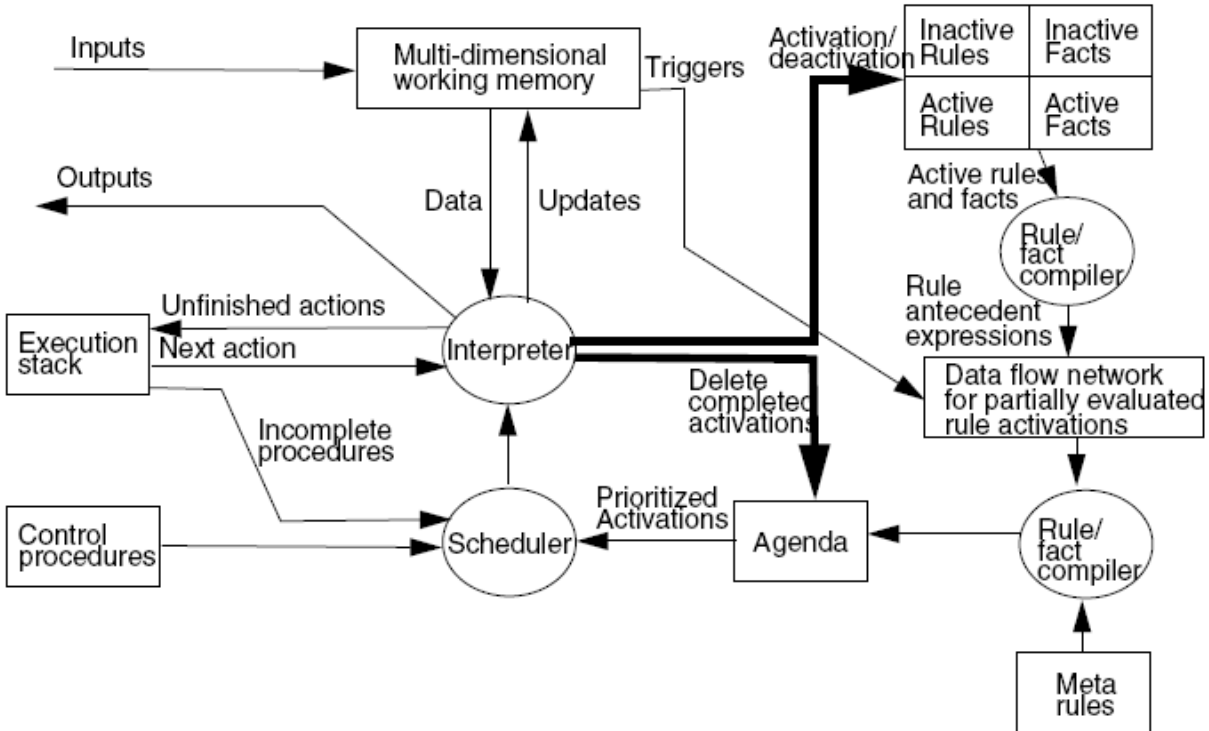
Donanım

- Küçük sayısal alt sistemlerin çok sayıda kopyası
- Alt sistemler derinlemesine analiz/test edilmiş.
- Tasarım hataları nadir



Yazılım

- Geniş ayrık durumlu alt sistemlerin az sayıda kopyası,tekrarlı yapı az
- Derinlemesine analiz/test edilmesi zor
- Doğru yapıyı bulmak zor



Yazılımda Kalite - 1

- Bilişim sektöründeki yazılım projelerinin maliyetlerinin bu denli büyümesiyle birlikte, yazılımda kalite kavramı günümüzde üzerinde en çok çalışılan konulardan biri haline gelmiştir.
- Crosby, genel olarak kaliteyi **“isterlere uygunluk”** olarak tanımlamıştır.

Yazılımda Kalite - 2

- Yazılım dünyasında kalite, kavram olarak herkesin hissedebildiği ancak neticede kimsenin tam olarak tanımlayamadığı soyut ve öznel bir olgu olarak karşımıza çıkmaktadır.
- Bir disiplin içerisinde kalitenin tam olarak tanımlanabilmesi için gelişmiş **ölçme** araçlarına ve sağlıklı karşılaştırmalar için tanımlı referans noktalarına gerek vardır.

Yazılımda Ölçme

- Ölçme kavram olarak varlıkların özelliklerinin sayısallaştırılması olarak tanımlanabilir. Bu noktada karşımıza bir yazılımda neleri sayılaştırabileceğimiz soruları çıkmaktadır.

"Ölçemediğimiz bir şeyi kontrol edemeyiz ve iyileştiremeyiz", (DeMarco, Tom)

Yazılım Kalite Sınıfları

- ISO 9126' ya göre yazılım kalite sınıfları:
  - İşlevsellik, (*Functionality*)
  - Güvenilirlik, (*Reliability*)
  - Kullanılabilirlik, (*Usability*)
  - Verimlilik, (*Efficiency*)
  - Bakılabilirlik, (*Maintainability*)
  - Taşınabilirlik, (*Portability*)

Yazılımın İç Özellikleri

- Bağımlılık, (*Coupling*):
  - İki sınıftan en az birinin diğerine etki etmesiyle oluşan bağımlılığın derecesidir.
- Uyumluluk, (*Cohesion*):
  - Sınıftaki metot ve niteliklerin birbirleriyle uyum dereceleridir.
- Karmaşıklık, (*Complexity*):
  - Sınıfların iç ve dış yapısını, sınıflar arası ilişkileri kavramadaki zorluğun derecesidir.

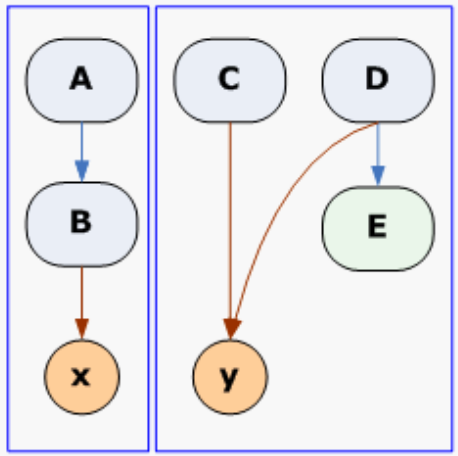
Yazılım Metrik Kümeleri

- Chidamber & Kemerer (**CK**) Metrik Kümesi
- Brito e Abreu **MOOD** Metrik Kümesi
- Bansiya ve Davis **QMOOD** Metrik Kümesi

Yazılım Metrikleri – (CK) Metrik Kümesi

- Sınıfın Ağırlıklı Metot Sayısı - Weighted Methods per Class (**WMC**)
- Kalıtım Ağacının Derinliği - Depth of Inheritance Tree (**DIT**)
- Alt Sınıf Sayısı - Number of Children (**NOC**)
- Nesne Sınıfları Arasındaki Bağımlılık - Coupling Between Object Classes (**CBO**)
- Sınıfın Tetiklediği Metot Sayısı - Response For a Class (**RFC**)
- Metotların Uyumluğu - Lack of Cohesion in Methods (**LCOM**)

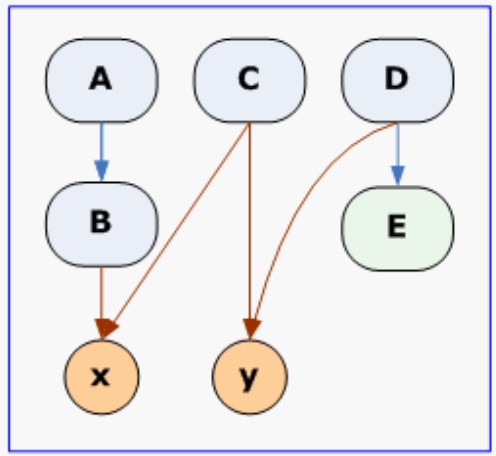
LCOM



LCOM4 = 2

The example on the left shows a class consisting of methods A through E and variables x and y. A calls B and B accesses x. Both C and D access y. D calls E, but E doesn't access any variables.

This class consists of 2 unrelated components (LCOM4=2). You could split it as {A, B, x} and {C, D, E, y}.



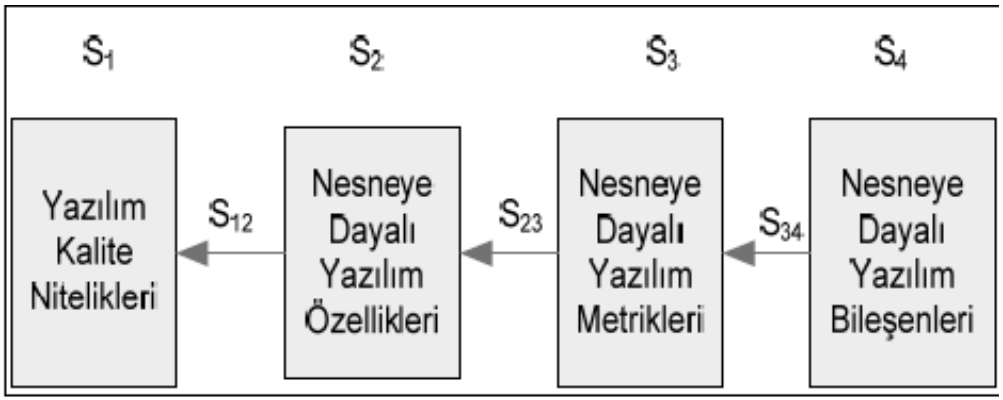
LCOM4 = 1

In the example on the right, we made C access x to increase cohesion.

Now the class consists of a single component (LCOM4=1). It is a cohesive class.

QMOOD- Metrik Kümesi

- Yazılım Kalite Nitelikleri: Bakılabilirlik,Taşınabilirlik
- Yazılım Özellikleri: Bağımlılık,Uyumluluk,Karmaşıklık
- Yazılım Metrikleri: CAM,NOM,ANA vs..
- Yazılım Bileşenleri: Paket,Sınıf, Metot
- Toplam Kalite indeksi



Düşük Kalite Göstergeleri (Bed Smells)

- Tekrar eden kod parçaları
- Uzun metotlar (Long Methods)
- Büyük sınıf (God Class)
- Uzun parametre listesi
- Farklı değişiklikler (Divergent Change)
- Parçacık tesiri (Shotgun Surgery)
- Veri sınıfı
- Reddedilen miras

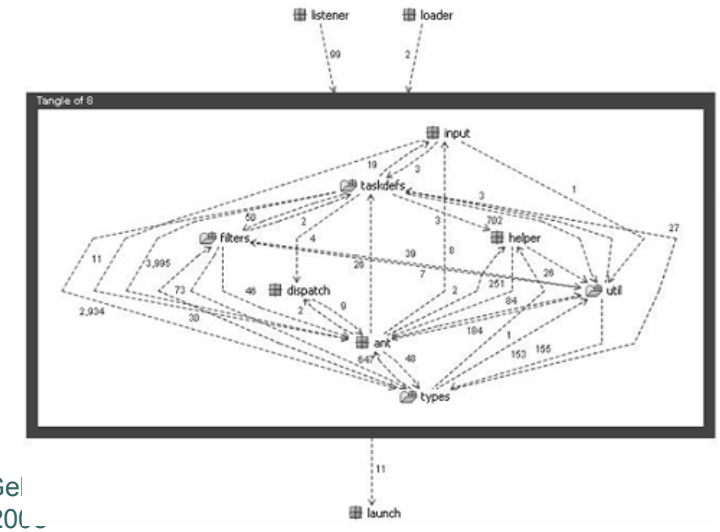
Tasarım Kalıpları

- **Yaratımsal Kalıplar**
  - Tekil Nesne (Singleton), Soyut Fabrika (Abstract Factory), İnşacı (Builder)…
- **Yapısal Kalıplar**
  - Adaptör, Köprü, Ön yüz (Facade)…
- **Davranışsal Kalıplar**
  - Gözlemci (Obsever), Strateji…

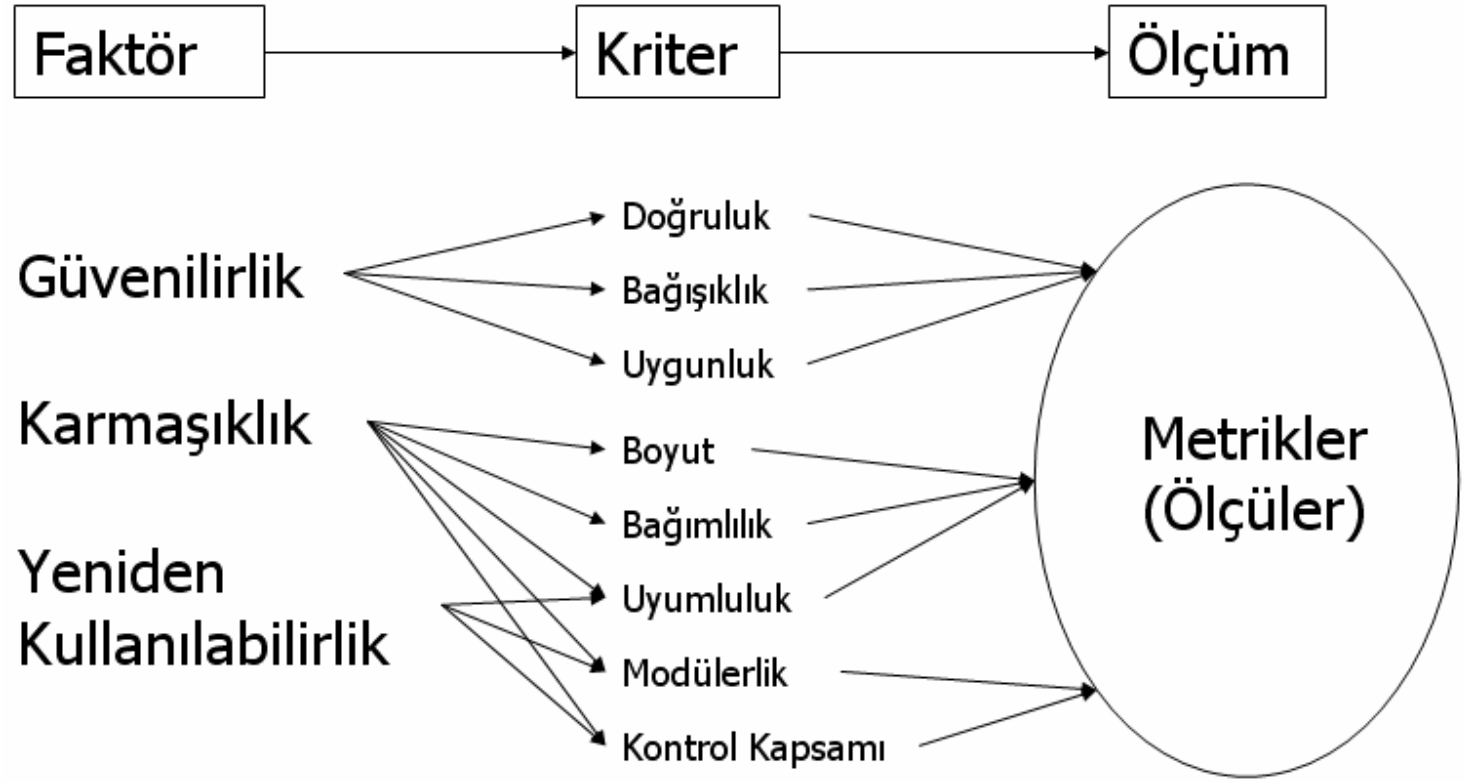
Yardımcı Araçlar

- **FindBugs,**
- **Metrics,**
- **PMD,**
- **Coverlipse,**
- **CheckStyle,**
- **SDMetrics,**
- **Coverity**

Metric	Total	Mean	Std. Dev.	Maximum
⊞ Number of Overridden Methods (avg/max per type)	11	2,2	2,926	8
⊞ Number of Attributes (avg/max per type)	9	1,8	2,713	7
⊞ Number of Children (avg/max per type)	3	0,6	1,2	3
⊞ Number of Classes (avg/max per packageFragment)	5	5	0	5
⊞ Method Lines of Code (avg/max per method)	206	6,438	10,105	40
⊟ Number of Methods (avg/max per type)	32	6,4	6,681	19
⊟ src	32	6,4	6,681	19
⊟ (default package)	32	6,4	6,681	19
⊞ SortItem.java	19	19	0	19
⊞ SortAlgorithm.java	7	7	0	7
⊞ QSortAlgorithm.java	4	4	0	4
⊞ BidirBubbleSortAlgorithm.java	1	1	0	1
⊞ BubbleSortAlgorithm.java	1	1	0	1
⊞ Nested Block Depth (avg/max per method)		1,688	0,982	4
⊞ Depth of Inheritance Tree (avg/max per type)		2,4	1,356	5

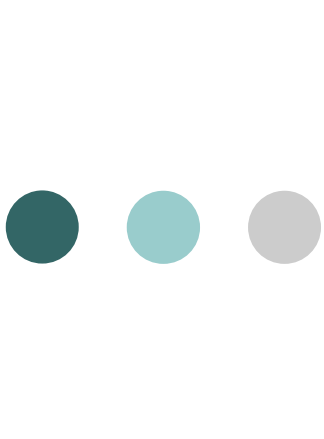


Faktör Kriter Ölçüm Modeli



Değerlendirme ve Sonuç

- Kaliteyi arttırmak her şeyden önce doğru ölçme yöntemlerine bağlıdır.
  - Doğru metriklerin tanımlanması, metriklerin doğru ölçülmesi ve doğru yorumlanması gerekir.
- Metrik tanımlamaları net ve ölçeklenebilir olmalı
  - CBO,RFC gibi…
- Yeni metrik tanımları kazandırılmalı
  - Örn. Okunurluk
- Metriklerin etkin kullanımı yaygınlaştırılmalı
  - Hata meyilli modüllerin belirlenmesi
  - Test ve Bakım maliyetlerinin öngörülmesi
- Metrikler, proje hedeflerine göre, bir bütün halinde değerlendirilmeli.
  - Metriklerin birbiriyle korelasyonu
- Yeni metrik araçları geliştirilmeli.
  - Dilden bağımsız, kolayca uyarlanabilir, görsel yanları güçlü
- Metrik ölçümleri yazılım geliştirme süreçlerine daha sıkı bağlanmalı.
  - Metriklerin değişimini izleme
  - Hataları erken giderme



Teşekkürler…