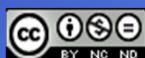


# YAZILIM TASARIMI KALİTESİ (ÖLÇME VE DEĞERLENDİRME)

Doç.Dr. Feza BUZLUCA  
İstanbul Teknik Üniversitesi  
Bilgisayar Mühendisliği Bölümü

<http://akademi.itu.edu.tr/buzluca>  
<http://www.buzluca.info>



Yazılım Tasarımı Kalitesi Ders Notlarının Creative Commons lisansı: CC BY-NC-ND 4.0

<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.tr>

BY: credit must be given to the creator.

NC: Only noncommercial uses of the work are permitted.

ND: No derivatives or adaptations of the work are permitted.

## Dersin Gerekçesi

- Her mühendislik süreci,
  - yapılan işin ve / veya elde edilen ürünün
  - Kalitesini (nedir?) hedeflenen (gerekli olan) düzeye
  - sadece gerektiği kadar kaynak kullanarak ullaştırmak için,  
ölçme, değerlendirme ve geri beslemeye (düzeltme, iyileştirme) gerek duyar.
- Yazılım geliştirme de bir mühendislik süreci olduğuna göre (yoksa daha çok zanaat mı, sanat mı?), ölçme (measurement), değerlendirme (assessment) ve iyileştirmeye (correction, improvement) gerek duyar.
- Yazılımın
  - işlevsel niteliklerinin test edilmesi (testing), doğrulanması (validation) ile
  - kalitesinin (örneğin, sürdürülebilirlik "maintainability") ölçülüp değerlendirilmesi (quality assessment)ortak noktaları olmakla birlikte farklı konulardır.

## Dersin Kapsamı

**Bu dersin ana konusu, yazılımların tasarım kalitesinin** (insan iş gücü gereksinimi en az düzeyde tutarak) nasıl ölçülebileceği (öngörülebileceği) ve bazı tasarım kusurlarının nasıl belirlenebileceğiidir.

Tasarım, yazılım geliştirme süreçlerinin ürünlerinden biridir.

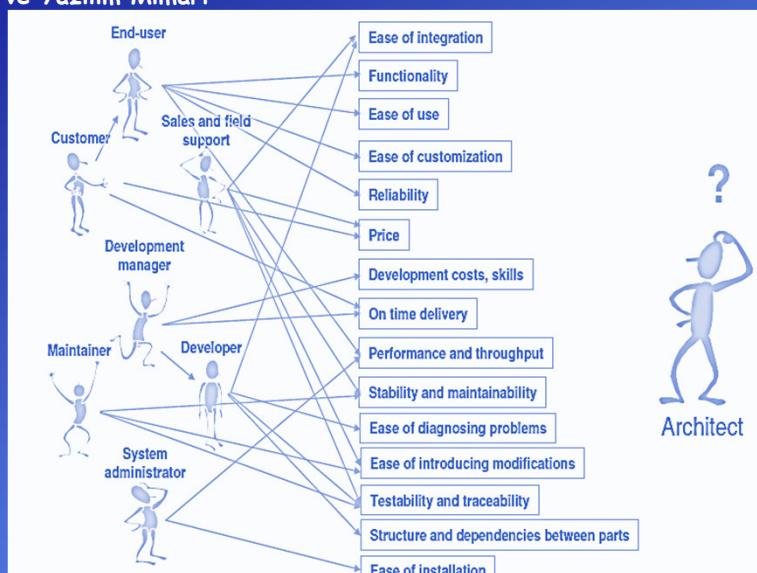
### Konular:

- Kalite modelleri
- Ölçme teorisi
- Yazılım metrikleri
- Tasarım kalite niteliklerinin ölçülmesi
- Tasarım kusurlarının belirlenmesi
  - Kural tabanlı yöntemler
  - Makine öğrenmesi temelli yöntemler
  - Büyük dil modellerinin (LLM) kullanılması
- Tasarım kopyalarının (*clone*) belirlenmesi
- Tasarımın görselleştirilmesi (*visualization*)

## Yazılım Kalitesi

- **Yazılım:** genellikle bir takım tarafından planlan, yönetilen ve yürütülen bir proje sonunda ortaya çıkan ürünlerin (analiz, tasarım, kod, dokümanlar) toplamıdır.
- **Yazılımın kalitesinin en genel tarifi** kendisinden beklenenleri ne kadar karşıladığıdır.
- Yazılım geliştirme ve sonuçta ortaya çıkan ürün birçok paydaşı (*stakeholder*) ilgilendiren geniş bir alandır ve bu paydaşların (rollerin) yazılımdan farklı bekentileri vardır (Bkz. Şekil yansısı 1.5).
- Yazılımın **değeri**, farklı **bekentileri (hedefleri)** sağlaması derecesi ile bunun için kullanılan **kaynaklar** (zaman, para, araçlar, vb.) arasındaki dengeye bağlıdır.  
 Az kaynakla bekentilerin sağlanması ürünün değerini artırır (mühendislik).
- Bir yazılımın değerlendirmesi farklı paydaşlara ve farklı bekentilere göre yapılabilir.  
 Bu nedenle bir yazılımı ölçüp değerlendirme,
  - somut olarak "iyi veya kötü bir yazılım" şeklinde kesin ve nesnel bir sonuç üreten teknik hesaplama işlemi değil,
  - öznellik (*subjectivity*), belirsizlik ve bazen de farklı bekentiler arasında tartışmalar içeren bir **karar alma** sürecidir.

### Yazılımdan Beklentiler ve Yazılım Mimarı



**Kaynak:** D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making techniques for software architecture design," *ACM Computing Surveys*, vol. 43, pp. 1-28, Oct. 2011.

<http://www.buzluca.info>

1.5

### Yazılım Kalitesi (devamı)

- Paydaşlar tarafından ölçülerek (bilinmek) istenen kalite nitelikleri çoğunlukla karmaşık yapıda olup birçok alt nitelikten oluşmaktadır.  
Örneğin sürdürülebilirlik (*maintainability*) özelliği; modülerlik (*modularity*), değiştirilebilme (*modifiability*), test edilebilme (*testability*) gibi alt kalite niteliklerinden etkilenmektedir.
- Ayrıca bu tür nitelikleri doğrudan ölçmek de çoğunlukla güçtür.  
Örneğin bir sistemin bakım kolaylığını doğrudan ölçmek için onu belli bir süre sahada çalışırmak ve bakımını yapmak (*test etmek*, hataları bulmak, değiştirmek vs.) gereklidir.
- Doğrudan, kısa sürede ölçülemeyen kalite niteliklerini değerlendirmek (*kestirmek*) için yazılımdan doğrudan elde edilebilecek uygun sayısal değerler (*metrikler*) kullanılır.  
Örneğin, bir yazılım sınıfındaki metot sayısını kullanıp hata çıkma sıklığını kestirmek gibi.
- Üst düzey kalite nitelikleri (*karakteristikler*) ile alt düzeydeki nitelikler ve metrikler arasındaki ilişkileri tanımlayan **kalite modelleri** ilerleyen bölgelerde ele alınacaktır.

## Yazılım ölçmesi neden gereklidir?

Yazılım dünyasında durum:

- Nerdeyse her elektronik cihazda yazılım var. Yazılım hayatımızı kontrol ediyor.
  - Yazılım hataları büyük zararlara (para, zaman, insan hayatı) neden olabilir.
  - Yazılımdan bekleniler (paydaş istekleri) artıyor ve karmaşıklığı artıyor.  
Buna bağlı olarak yazılımların yapısı da karmaşıklığı artıyor, boyutları artıyor.  
Bu tür yazılımlar hatalara açıktır. İnsan gözüyle kalite değerlendirmesi yapmak zordur.
  - Önlem alınmadığında yazılım (özellikle bakım) maliyetleri çok artıyor.  
Güncelleme, uyarlama, hata düzeltme, yeniden kullanma maliyetleri yüksektir.
  - Sektörde rekabet var (zaman, maliyet).
  - Aynı işi yapan satın alabilecek birden fazla yazılım çözümü var.
- Sonuç olarak hem yazılım geliştiren firmalar hem de yazılımı satın alan müşteriler ölçmeye gerek duyarlar.
- Yazılım firmaları, zaman ve bütçe kısıtlarına uyarak paydaş beklenilerini karşılamak ve rekabetçi pazarda yer alabilmek için kalite kontrolüne gerek duyar.
  - Yazılımın müşterileri satın alacakları yazılımı seçmek ve kullandıkları yazılıma güvenebilmek için ölçmeye gerek duyar.

## Yazılım ölçmesi hangi amaçlarla kullanılır?

- **Hedefleri belirleme:** Başlamadan önce projenin ölçülebilir, somut başarı kriterleri belirlenir.
  - **Durumu belirleme ve karar verme:** Ölçme ile karar vericilerin (tasarımcı, proje lideri, vd.) karşılaşışı belirsizlikleri azaltmak hedeflenir.
- Yazılım firması yaptığı işin beklenileri ne kadar karşıladığı (kalite düzeyini, kaynak tüketimini) ve nasıl planlama yapması gerektiğini belirleyebilir.
- Örneğin; bu gereksinimlere göre projeyi zamanında bitirmek için ne kadar iş gücü (adam x ay) gereklidir?
  - Beklenen düşük hata sayısına ulaşmak için testleri daha ne kadar sürdürmek gereklidir?
  - Yeni sürüm (veya son ürün) çıkmaya hazır mı? Tasarım kalitesi düzeyi yeterli mi?
- **İyileştirme:** Firma yaptığı işi iyileştirebilecek veriler elde edebilir.
  - Bir proje devam ederken erken aşamalarda nasıl iyileştirilebilir?
  - Projenin (yazılımının) hangi bölümünde, ne tür kusurlar var?

### Yazılım ölçmesi hangi amaçlarla kullanılır? (devamı)

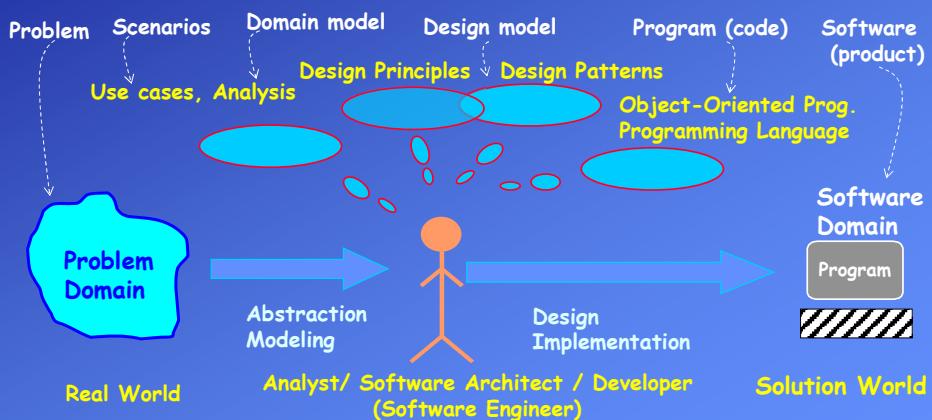
- **Kanıtlama:** Firma, müşterilerine ürünün kaliteli olduğunu somut bir biçimde gösterebilir.
  - Müşteri aldığı yazılıma nasıl güvenebilir? Kalite puanlaması yapılabılır mı?
- **Kestirim/öngörü:** Ölçme ile yazılımın o anki durumu belirlendiği (*assessment /benchmarking*) gibi, sonraki sürümler (ya da projeler) hakkında kestirimler de (*estimation*) yapılabilir.
  - İleride sorun/hata çıkma olasılığı yüksek olan birimler hangileridir, sorunları nedir (*prediction*)?
- **Karşılaştırma:**
  - Yeni sürüm daha mı iyi?
  - Şimdiki ürünü/sürümü değiştirme zamanı geldi mi?
  - Müşteri hangi yazılımı alacağına nasıl karar verebilir?
- Bir yazılım projesinde süreç, ürün ve kaynak kalitesi ölçülebilir.

Bu derste, yazılım projesinin temel ürünlerinden olan **tasarımin kalitesi** üzerinde durulacaktır.

### Tasarım Kalitesi ve Önemi

#### Hatırlatma:

- Bir yazılımın geliştirilmesi sürecinde yer alan **tasarım** aşamasında
  - Yazılım sistemini oluşturan birimler (sınıf, modül vb.) belirlenir.
  - Bu birimlerin sorumlulukları (hizmetleri) belirlenir.
  - Birimler arası ilişkiler (bağlantılar) belirlenir.



## Tasarım Kalitesi ve Önemi (devamı)

Hatırlatma:

- Bir yazılım projesinin toplam maliyetinin (zaman, iş gücü) sadece %10-%15'i tasarım aşamasında harcanmaktadır.
- Tasarımdan kaynaklanan hataları düzeltmek için bakım aşamasında harcanan miktar proje maliyetinin %80' e ulaşabilmektedir.

Tasarım kalitesinin bir tanımı:

- Tasarım, kodlama, test, hata giderme ve bakım maliyetlerini düşük tutan bir tasarım, **iyi bir tasarım**dır.  
P. Coad and E. Yourdon. Object-Oriented Design. Prentice Hall, London, 2/e, 1991.

Diger bir tanım:

- **Yüksek kaliteli bir tasarım** şu özelliklere sahip kaliteli ürünler oluşturmalıdır:  
Kolay anlaşılır, kolay gerçekleştirilebilir, kolay sızanır, kolay değiştirilebilir.  
S.L. Pfleeger. Software Engineering - Theory and Practice. Prentice-Hall, 1998.

Tasarımın Önemi:

- Yazılımın tasarım kalitesi, birçok kalite niteliğini doğrudan etkilemektedir.
- Yazılımın doğru çalışması ve güvenilir olması tasarım kalitesine bağlıdır.
- Yazılımın maliyetinin düşük olması (bakım, tekrar kullanılabilirlik) tasarım kalitesine bağlıdır.

## Yazılım mühendisliğinde ölçmenin göz ardı edilmesi ve sonuçları

- Mühendislik çalışmalarında ölçme olmadan tasarımlar yapılması ve ürünler oluşturulması düşünülemez.  
Örneğin; bir elektrik devresi tasarlarken kullanılacak olan elemanların özellikleri, beklenen hedeflere ve belli kurallara (örneğin Ohm kanunu) göre baştan belirlenir.  
Devre gerçekleştirildikten sonra da ölçme işlemleri ile devrenin akım, güç tüketimi gibi özellikleri ölçülür, istenen hedeflere ulaşılıp ulaşmadığı görülür.
- Yazılım dünyasında ise ölçme hala ikinci planda kalmaktır; ölçme yapmadan yüksek kaliteli (verimli, güvenilir vb.) ürünler oluşturulması beklenmektedir.

Ölçmenin göz ardı edilmesi nedeniyle günümüzde birçok projede rastlanan **sorunlar**:

- **Projenin başında** yazılım ürünlerine ilişkin somut, ölçülebilir hedefler oluşturulamamaktadır.  
Örneğin; kolay kullanılır, bakım kolaylığı yüksek ve güvenilir bir yazılım oluşturulacağı iddia edilir.  
Ancak bu terimlerin ne anlama geldiği ve nasıl ölçüleceği açıkça belirlenmediği (bilinmediği) için yazılım geliştirilirken bu hedeflere uygun şekilde çalışmak mümkün olmamaktadır.  
Proje sonunda da bu hedeflere ulaşılıp ulaşmadığı belirlenmemektedir.

### **Yazılım mühendisliğinde ölçmenin göz ardı edilmesinin sonuçları (devamı)**

Ölçmenin göz ardı edilmesi nedeniyle günümüzde birçok projede rastlanan sorunlar:

- **Proje devam ederken** ölçme yapılmadığından bazı hataları erken aşamalarda düşük maliyetlerle gidermek mümkün olmamaktadır.
  - Örneğin; ekipteki bazı yazılımcılar değiştirilmesi zor, karmaşık kodlar yazıyor olabilirler.
  - Bu tip modüller çalışma sırasında hata çıkardıkları için geliştirme aşamasının birçok sürümünde hata çıkarmayabilirler.
  - Ancak bir değişiklik veya ekleme yapmak gerekiğinde bu modüller büyük maliyetlere neden olmaktadır.
  - Diğer bir sorunda ekipte değişiklik olduğunda yeni katılanların bu modüller üzerinde çalışmamasıdır.
- **Üretilen ürünün o andaki veya gelecekteki kalitesi** hakkında somut bir öngörüde bulunulamamaktadır.
  - Örneğin; potansiyel bir müşteriye ürünün güvenilirliği hakkında (belli bir sürede tahmini kaç hata çıkar) bilgi vermek mümkün olmamaktadır.
  - Örneğin; yazılım firması bir ürünü başka bir donanıma taşımadan maliyetini kestirememektedir.

### **Yazılım (Tasarım) Kalitesinin Ölçülmesindeki Zorluklar**

1. Yazılımlar tam olarak somut varlıklar değildir.  
 Tasarımlar (sadece yazılımda değil) sezgisel olarak algılanabilen ancak sayısal olarak ifade edilmesi güç olan (sanatsal?) özellikler içerirler.
2. Ölçme teorisinin yeteri kadar anlaşılmadan (ya da doğru şekilde uygulanmadan) yazılım kalitesinin ölçülmeye çalışılması hatalı sonuçlar üretiyor.
3. Yazılımın dış kalite niteliklerinin karmaşık yapıda olması ve proje tamamlanmadan doğrudan ölçülememesi.  
 Hangi kalite niteliğinin, hangi iç özelliklere, ne kadar bağlı olduğu da net değildir.  
 Örneğin tekrar kullanılabilirlik hangi iç özelliklerden hangi oranda etkilenir?



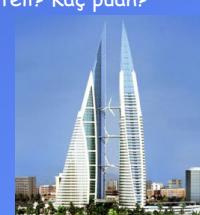
Mona Lisa  
Da Vinci



Skull and Pitcher  
Picasso



Londra Gherkin Tower



Bahreyn  
Dünya Ticaret Merkez.

### Yazılım (Tasarım) Kalitesinin Ölçülmesindeki Zorluklar (devamı)

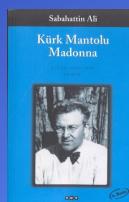
Yazılımın kaynak kodu bir yörenyle uzun bir metin olarak düşünülebilir.

Bu metnin yapısal kalitesi nasıl sayıya dönüştürülür (ölçülür)?

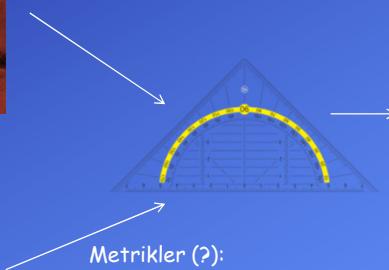
Oğuz Atay  
Tutunamayanlar, 1972  
736 s.



Sebahattin Ali  
Kürk Mantolu Madonna,  
1943, 163 s.



Ölçülecek Varlıklar



Ölçme

**Değerlendirme:**  
(Yapılabilir mi?)

- Kurgusu iyidir (8/10 dur).
- Karakterler gerçekcidir.
- A, B'den daha iyidir.
- Öykü heyecanlıdır (75/100).
- Anlaşılmazı zordur (40/100).

Değerlendirme

### Yazılım Ölçmesi Çalışmalarında Durum

- Yazılım kalitesi doğası nedeniyle somut olarak değerlendirilmesi zor bir kavramdır.  
Örneğin; sürdürilebilirlik (*Maintainability*), güvenlilik, bakım kolaylığı nasıl ölçülür ve öngörülür?
- Bu nedenle yazılım ölçmesi, tüm gereksinimleri karşılayan, olgunlaşmış yöntemlere sahip bir alan değildir.
- Bununla birlikte, duyulan gereksinim ve konunun önemi nedeniyle, bu konuda birçok araştırma yapılmakta ve bazı eksiklikleri olsa da çeşitli yöntemler (*modeller*) ve araçlar geliştirilmektedir.
- Geliştirilen yöntemler büyük ölçüde görgülü (*empirical*) çalışmalara dayanmaktadır.
- Yazılım mühendisliğinin dayandığı matematik ve ölçme teorisi diğer mühendislik disiplinlerine göre daha yenidir (1980'lerden sonra).
- Araştırmalar devam ettiğinden güncel durumu bilmek için bilimsel yayınları takip etmek önemlidir.