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 Feza BUZLUCA'ya aittir.
Lisans: https://creativecommons.org/licenses/by-nc-nd/4.0/deed.tr

Giriş

Her mühendislik süreci,
yapılan işin ve / veya elde edilen ürünün kalitesini gerekli olan (beklenen) düzeye sadece gerektiği kadar kaynak kullanarak ulaştırmak 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.

Bir yazılım ürünü oluşturulurken veya ürün ortaya çıktıktan sonra yazılım ile ilgili aşağıdaki üç grupta yer alan farklı varlıklar (*entity*) ile ilgili ölçmeler yapılabilir.

1. Süreçler (*Processes*): Projenin yönetimi; zamanlama, hedeflere ulaşma
2. Ürünler (*Products*): Yazılım geliştirme süreçlerinde oluşan her türlü ürün; gereksinim raporu, tasarım (diyagramlar), kod, doküman
3. Kaynaklar (*Resources*): Süreçlerde kullanılan her türlü kaynak; ekibin verimliliği, kullanılan araçların (derleyici, kütüphane) verim, ofis

Yazılımın (işlevsel niteliklerinin) test edilmesi ile yazılım kalitesinin ölçülmesi farklı konulardır.

Dersin Kapsamı

Bu dersin ana konusu, nesneye dayalı (*object-oriented*) yazılımların tasarım kalitesinin nasıl ölçülebileceği (öngörülebileceği) ve bazı tasarım kusurlarının nasıl belirlenebileceğidir.

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
- Tasarım kopyalarının (*clone*) belirlenmesi
- Tasarımın görselleştirilmesi (*visualization*)

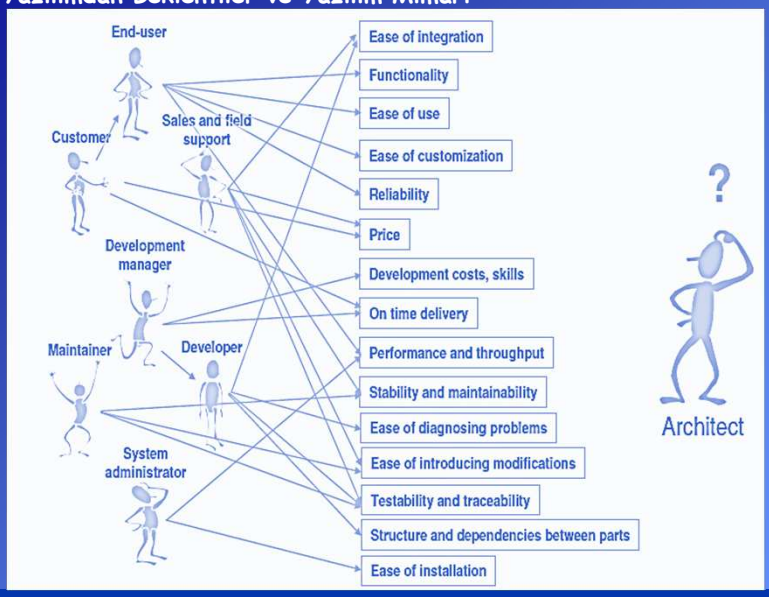
2012-2023 Feza BUZLUCA

Yazılım Kalitesi

- **Yazılım**; genellikle bir takım tarafından planlan, yönetilen ve yürütülen bir proje sonunda ortaya çıkan ürünlerin (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 beklentileri vardır (Bkz. Şekil yansı 1.5).
- Yazılımın **değeri**, farklı beklentileri sağlama derecesi ile bunun için kullanılan kaynaklar (zaman, para, araçlar, vb.) arasındaki dengeye bağlıdır.
  Az kaynakla beklentilerin sağlanması ürünün değerini artırır (mühendislik).
- Bir yazılımın değerlendirmesi farklı paydaşlara ve farklı beklentilere göre yapılabilir.
  Bu nedenle bir yazılımı ölçüp değerlendirme, "somut olarak iyi bir yazılım" şeklinde kesin ve nesnel bir sonuç üreten teknik hesaplama işlemi değil, öznellik (*subjectivity*), belirsizlik ve bazen de farklı beklentiler arasında çatışmalar içeren bir karar alma sürecidir.

2012-2023 Feza BUZLUCA

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.

Yazılım Kalitesi (devamı)

Paydaşlar tarafından ölçülmek (bilinmek) istenen kalite nitelikleri çoğunlukla karmaşık yapıda olup birçok alt nitelikten oluşmaktadır.

Örneğin bakım kolaylığı (*maintainability*) özelliği; modülerlik (*modularity*), değiştirilebilme (*modifiability*), test edilebilme (*testability*) gibi alt kalite niteliklerinden etkilenmektedir.

Ayrıca bu tür nitelikleri doğrudan ölçmek de çoğunlukla mümkün değildir.

Örneğin bir sistemin bakım kolaylığını doğrudan ölçmek için onu belli bir süre sahada çalıştırmak ve bakımını yapmak (test etmek, hataları bulmak, değiştirmek vs.) gerekir.

Doğrudan, kısa sürede ölçülemeyen kalite niteliklerini değerlendirmek (kestirmek) için yazılımdan doğrudan elde edilebilecek uygun sayısal değerler (metrikler) kullanılır.

Örneğin, bir yazılım sınıfındaki metot sayısı gibi.

Üst düzey kalite nitelikleri (karakteristikler) ile alt düzeydeki nitelikler ve metrikler arasındaki ilişkileri tanımlayan **kalite modelleri** ikinci bölümde ele alınacaktır.

2012-2023 Feza BUZLUCA

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 beklentiler (paydaş istekleri) artıyor ve karmaşıklaşıyor.
  Buna bağlı olarak yazılımların yapısı da karmaşıklaşıyor, 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
- Sektörde rekabet var (zaman, maliyet).
- Aynı işi yapan satın alınabilecek 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 kısıtlarını aşmayarak istekleri 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.

2012-2023 Feza BUZLUCA

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.) karşılaştığı belirsizlikleri azaltmak hedeflenir.
  Yazılım firması yaptığı işin beklentileri ne kadar karşıladığını (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) gerekir?
  - Beklenen düşük hata sayısına ulaşmak için testleri daha ne kadar sürdürmek gerekir?
  - 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 (yazılımın) hangi bölümünde, ne tür kusurlar var?

2012-2023 Feza BUZLUCA

Yazılım ölçmesi hangi amaçlarla kullanılır? (devamı)

- *Kanıtlama*: Firma, müşterilerine ürünün kaliteli olduğunu somut bir biçimde gösterebilir.
  - Müşteri aldığı yazılıma nasıl güvenebilir? Kalite puanlaması yapılabilir mi?
- *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 çıkarma 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 **tasarımın kalitesi** üzerinde durulacaktır.

2012-2023 Feza BUZLUCA

Tasarım Kalitesi ve Önemi

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 tasarımdır**.

P. Coad and E. Yourdon. Object-Oriented Design. Prentice Hall, London, 2 e, 1991.

Diğer bir tanım:

- **Yüksek kaliteli bir tasarım** şu özelliklere sahip kaliteli ürünler oluşturmalıdır: Kolay anlaşılır, kolay gerçeklenir, kolay sınanır, kolay değiştirilebilir.

S.L. Pfleeger. Software Engineering - Theory and Practice. Prentice-Hall, 1998.

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.

2012-2023 Feza BUZLUCA

Yazılım mühendisliğinde ölçmenin göz ardı edilmesi ve sonuçları

Mühendislik çalışmalarında ölçme olmadan tasarımlar yapılması ve ürünler oluşturulması düşünülemez.

Örneğin; bir elektrik devresi tasarlanırken kullanılacak olan elemanların özellikleri, beklenen hedeflere ve belli kurallara (örneğin Ohm kanunu) göre baştan belirlenir.

Devre gerçekleştirildikten sonra da ölçme işlemeleri ile devrenin akım, güç tüketimi gibi özellikleri ölçülür, istenen hedeflere ulaşılıp ulaşılmadığı görülür.

Yazılım dünyasında ise ölçme hala ikinci planda kalmakta; ölçme yapmadan yüksek kaliteli (verimli, düşük maliyetli, 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 ulaşılamadığı belirlenememektedir.

2012-2023 Feza BUZLUCA

Yazılım mühendisliğinde ölçmenin göz ardı edilmesinin sonuçları (devamı)

Ölçmenin göz ardı edilmesi nedeniyle günümüzde birçok projede rastlanan sorunlar:

- 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 taşımanın maliyetini kestirememektedir.

- 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 çıkarmadıkları için geliştirme aşamasının birçok sürümünde hata çıkarmayabilirler.

  Ancak bir değişiklik veya ekleme yapmak gerektiğinde bu modüller büyük maliyetlere neden olmaktadırlar.

  Diğer bir sorunda ekipte değişiklik olduğunda yeni katılanların bu modüller üzerinde çalışamamasıdır.

2012-2023 Feza BUZLUCA

Yazılım (Tasarım) Kalitesinin Ölçülmesindeki Zorluklar

1. Yazılımlar tam olarak somut varlıklar değiller.
   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 kalite niteliklerinin birden fazla alt bileşenden oluştukları için doğrudan ölçülememesi.
   Hangi kalite niteliğinin, hangi iç özelliklere, ne kadar bağlı olduğu da net değildir.
   Örneğin tekrar kullanılabilirlik hangi iç özelliklerden hangi oranda etkilenir?

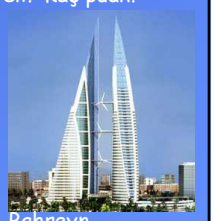
Hangisi daha güzel? Kaç puan?

Mona Lisa
Da Vinci

Skull and Pitcher
Picasso

Hangi tasarım daha kaliteli? Kaç puan?

Londra Gherkin Tower



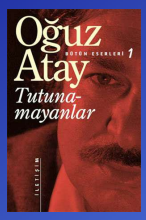
Bahreyn
Dünya Ticaret Merk

2012-2023 Feza BUZLUCA

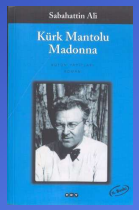
Yazılım (Tasarım) Kalitesinin Ölçülmesindeki Zorluklar (devamı)

Yazılımın kaynak kodu bir yönüyle uzun bir metin olarak düşünülebilir.
Bu metnin yapısal kalitesi nasıl sayıya dönüştürülür (ölçülür)?

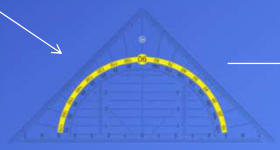
Oğuz Atay
Tutunamayanlar, 1972
736 s.



Sebahattin Ali
Kürk Mantolu Madonna,
1943, 163 s.



Ölçülecek Varlıklar



Metrikler (?):
Satır sayısı
Romandaki karakter sayısı
Karakterler arası ilişki sayısı

Ölçme

Değerlendirme:
(Yapılabilir mi?)

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

Değerlendirme

2012-2023 Feza BUZLUCA

Nasıl bir ders?

Yazılım mühendisliğinin dayandığı matematik ve ölçme teorisi diğer mühendislik disiplinlerine göre çok yenidir (1980'lerden sonra).

Ayrıca, yazılım kalitesi karmaşık doğası nedeniyle somut olarak değerlendirilmesi zor bir kavramdır. Örneğin; metot karmaşıklığı, bakım kolaylığı nasıl ölçülür?

Bu nedenle yazılım ölçmesi, tüm gereksinimleri karşılayan, olgunlaşmış ve yaygın kullanımı olan 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 görgül (*empirical*) çalışmalara dayanmaktadır.

Bu derste, geçerliliği herkes tarafından kabul edilen, yaygın kullanılan, tamamen olgunlaşmış ölçme ve değerlendirme yöntemleri anlatıl(a)mayacaktır.

Derste, yazılım tasarımının ölçülmesi konusunda yapılan araştırmalar ve geliştirilen deneysel yöntemler, geçmişten günümüze doğru tanıtılacak, önerilen yöntemler **öğrenciler ile birlikte incelenerek karşılaştırılacaktır**.

Her hafta okunması gereken dokümanlar öğrencilere bildirilecektir.

2012-2023 Feza BUZLUCA

Bu dersi almalı mıyım?

1. Bu dersten önce nesneye dayalı yazılım tasarımı yöntemlerinin anlatıldığı bir ders almış olmanız gerekir.

   Örneğin:

   - Object-Oriented Modeling and Design
     http://ninova.itu.edu.tr/tr/dersler/bilgisayar-bilisim-fakultesi/2097/blg-468e/

   Ayrıca aşağıdaki dersleri almış olmanız büyük ölçüde yararlı olur:

   - Statistics and Estimation in Computer Science
   - Machine Learning

2. Bu derste programlama öğretilmemektedir.

   Bu ders daha çok, yazılım tasarımı kalitesi konusunda araştırma ve tez yapmak isteyen öğrenciler için uygundur.

2012-2023 Feza BUZLUCA

Nesneye Dayalı Tasarım (*Object Oriented Design*)

Temel Kavramlar (anımsatma):

- *Data Abstraction* (Soyutlama):

Gerçek dünyadaki varlıkların soyut modelleri oluşturulur. (Sınıf yapısı ile)

Bu modeller ilgili varlığın sadece yazılımda gerekli olan özelliklerini ve davranışlarını (hizmetlerini) içerirler.

Bu varlıklar hizmet alan-veren (*client-server*) mantığı ile ilişkide bulunurlar.

- *Encapsulation* (Kapsülleme):

Soyut modeller oluşturulurken ilgili veriler ve metotlar aynı birimin içine konur.

Diğer birimleri ilgilendiren hizmetler açık (*public*) olarak tanımlanırken modelin iç yapısı (*implementation*) dış dünyaya açılmaz (*private*). "*Data hiding*"

- *Inheritance* (Kalıtım): *"is a" relation*

Sınıflar arasında genel / özel ilişkisi oluşturulur (*generalization /specialization*).

- *Polymorphism* (Çok şekillilik):

Aynı arayüze (*interface*) sahip olan nesneler sistemin çalışması sırasında birbirilerinin yerine geçebilirler.

O andaki koşullara göre uygun olan nesne etkin olur ve ilgili hizmeti yerine getirir.

Hizmeti alan nesne bunun bilincinde olmak zorunda değildir.

İlgili tasarım prensibi: "*Design to interface not to implementation.*"

2012-2023 Feza BUZLUCA

Nesneye Dayalı Tasarımın Kalitesi

Nesneye dayalı yöntemden beklenen temel tasarım kalitesi nitelikleri (*quality attributes*):

- Understandability (Anlaşılırlık): Ekipteki değişimler, yeni gelenlerin tasarımı çabuk kavraması
- *Modularity* (Modülerlik): Bağımsız geliştirilebilen, sınanabilen birimlerin olması
- *Reusability* (Tekrar kullanılabilirlik): Maliyet düşürme
- *Testability* (Sınama kolaylığı): Hata çıktığında yerinin kolay bulunması ve kolay giderilebilmesi
- *Reliability* (Güvenilirlik): Hataya açık olmaması, hataların erken aşamada bulunmaları
- *Flexibility, Modifiability, Extensibility* (Esneklik): Değişen isteklere kolay uyum sağlama, yeni özelliklerin kolay eklenebilmesi.

Sonuç: *Maintainability* (Bakım kolaylığı)

Derste ele alınacak problem:

Bir yazılımda bu niteliklerin olup olmadığı veya ne düzeyde olduğu nasıl ölçülür?

Bu nitelikler nasıl anlamlı sayılara dönüştürülür?

2012-2023 Feza BUZLUCA

Nesneye Dayalı Tasarımın Kalitesi (devamı)

Anımsatma: Nesneye dayalı bir yazılım kaliteli olmasını garantileyen genel, somut kurallar yoktur.

Sezgiler ve deneyim önemlidir.

Yine de yıllar içinde deneyimle oluşturulmuş prensipler ve kriterler vardır.

Buna göre iyi bir nesneye dayalı tasarımın iç özellikleri aşağıdaki gibi olmalıdır:

- ***Manageable Complexity* (Yönetilebilir karmaşıklık):**

Yazılımların karmaşıklığı sınırlı tutulabilmeli.

(Karmaşıklığın tanımı ve ölçülme yöntemi yazılım dünyasında tartışmaya açıktır.)

Karmaşıklığı düşük olan yazılımların kolay anlaşılabilir ve kolay değiştirilebilir olduğu varsayılır.

- ***Low coupling* (az bağımlılık):**

Sınıflar arası bağımlılık (etkileşim) sınırlı olmalı.

Bir sınıftaki değişim diğerlerini etkilememeli.

Bir sınıf diğerlerinden bağımsız olarak anlaşılabilmeli.

Bir sınıftaki hata diğer sınıfları etkilememeli.

2012-2023 Feza BUZLUCA

- ***High Cohesion* (İyi uyum):**

Sınıf iyi tanımlanmış belli bir amaç için oluşturulmalı.

Sınıfın üyeleri birbirleri ile ilgili olmalı.

Bağımlılık ile iyi uyum arasındaki denge o yazılımın modülerliğini belirler.

Uyum ile ilgili bazı kurallar:

Bir sınıf gerçek dünyadaki sadece bir varlığa karşı düşmeli (*A class should capture one and only one abstraction*).

İlgili veriler ve davranışlar aynı yerde bulunmalı.

Bir metot sadece bir iş yapmalı.

- ***Proper Data Abstraction* (Uygun Soyutlama):**

Gerçek dünya uygun şekilde modellenmeli.

Gerçek dünyadaki varlıklar uygun şekilde sınıflara bölünmeli.

Sınıflar arası hizmet alma ilişkileri (işbirliği) uygun olmalı.

Sınıflar arası kalıtım hiyerarşisi uygun olmalı.

Aslında bundan önce sayılan tüm özellikler uygun soyutlamanın (modellemenin) sonuçlarıdır.

2012-2023 Feza BUZLUCA

Kaliteli Tasarım oluşturmakta yol gösteren prensip ve kalıplar

Tasarım prensipleri ve kalıpları doğru şekilde uygulandığında projedeki zaman baskısı ve isteklerin sık değişmesinin yarattığı sorunlar da azalmaktadır.

Nesneye dayalı tasarımın önemli prensip ve kalıplarını "Object Oriented Modeling and Design" dersinin ders notlarından hatırlayınız.
http://ninova.itu.edu.tr/tr/dersler/bilgisayar-bilisim-fakultesi/2097/blg-468e/

Örnek prensipler ve kalıplar:

- Separation of concerns
- Model-View separation
- Low Coupling
- High cohesion
- Favor object composition over class inheritance
- Design to interface and not concrete classes
- Adapter (GoF)
- Factory
- Strategy (GoF)
- Composite (GoF)
- Decorator (GoF)

Okunacak doküman: Adı geçen konuları "Object Oriented Modeling and Design" dersinin ders notlarında gözden geçiriniz.

2012-2023 Feza BUZLUCA

Kaynak Kitaplar:

Ölçme Teorisi:

- Fenton, N and Bieman, J. (2015), Software metrics: A rigorous and practical approach, 3/e, CRC Press.

Genel bilgiler:

- Ruchika Malhotra (2016), Empirical Research in Software Engineering, CRC.
- Abran, A. (2010), Software Metrics and Software Metrology, IEEE Computer Society- John Wiley&Sons, Hoboken, NJ, USA.

Nesneye Dayalı Tasarım Kusurları:

- Lanza. M. and Marinescu, R. (2006), Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems, Springer -Verlag, Berlin.

Notlandırma:

- Yıl içi sınavı: %30
- Ödevler, Proje ve Sunum: %30
- Yıl sonu: %40

Yarıyıl Sınavına Girme Koşulu: Yarıyıl içi notu (sınav ve proje ortalaması) en az 50 olmalı.

2012-2023 Feza BUZLUCA