

**YAZILIM MÜHENDİSLİĞİ TEMELLERİ – Şubat 2017**  
**Yrd.Doç.Dr. Yunus Emre SELÇUK**  
**GENEL BİLGİLER**

**BAŞARIM DEĞERLENDİRME**

- 1. Ara sınav: 13/04/2017, yazılı, analiz konusu dahil.
- 2. Ara sınav: 11/05/2017, test
- Final sınavı: Final haftasında, tasarım ağırlıklı
- Proje ödevi: İki kişilik ekipler ile, son ders haftasında sunum yapılacaktır.
- **Sınav ve sunum tarihleri dersin ilerleyişine göre değişebilir.**
- Puanlama (değişebilir):
  - 1. Vize \* %20, 2. Vize \* %20, Final \* %40, Proje \* %20

**KAYNAK KİTAPLAR**

- Software Engineering / Ian Sommerville. Addison-Wesley, 2010, 9<sup>th</sup> ed.
- Software Engineering: A Practitioner's Approach / Roger S. Pressman. McGraw/Hill, 2005, 6<sup>th</sup> ed.
- UML Distilled / Martin Fowler. Addison Wesley, 2003, 3<sup>rd</sup> ed.
- Applying UML and Patterns: Intro. OOAD & Iterative Development / Craig Larman. Prentice-Hall, 2004, 3<sup>rd</sup> ed.
- ... ve diğerleri

1

**GENEL BİLGİLER**

**DERS İÇERİĞİ**

- Yazılım Mühendisliğine Giriş
- Yazılım Geliştirmede Süreç Modelleri
- Gereksinim Mühendisliği
- Nesneye Yönelik Çözümleme
- Nesneye Yönelik Tasarım
- Yazılım Ölçütleri
- Yazılım Sınama Teknikleri
- Yazılım Proje Yönetimine Giriş

2

## YAZILIM MÜHENDİSLİĞİ TEMELLERİ DERS NOTLARI

Yrd.Doç.Dr. Yunus Emre SELÇUK

### YAZILIM MÜHENDİSLİĞİNE GİRİŞ

3

### YAZILIM MÜHENDİSLİĞİNE GİRİŞ

#### YAZILIM

- Yazılım :
  - Herhangi bir boyuttaki herhangi bir tür donanımda çalışan bilgisayar programını VE,
  - Basılı veya elektronik ortamdaki her tür dokümanı içeren ürün.
    - Dokümanlar yazılım geliştirme ve son kullanıcıya yönelik olabilir.
- Yazılım bir üründür, ancak başka ürünler geliştirmeye veya elde etmeye yarayan bir araç da olabilir.
- Yaşam döngüsü: Yazılımın bir fikir olarak doğmasından, kullanım dışı bırakılmasına kadar geçen süreç.
- Yazılım fiziksel bir ürün olmadığı için aşınmaz, ancak zamanla yetersizleşebilir.
  - Değişim kaçınılmazdır: Yazılım, yaşam döngüsü süresince değişikliklere uğrar.
  - Değişiklikler, yazılımda yeni hatalar oluşturabilir.
  - Yeni hatalar tam olarak düzeltilmeden yeni değişiklikler gerekebilir.
- Çözüm: Yazılım mühendisliği ilkelerine uyularak daha iyi tasarlanmış yazılım.

4

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM TÜRLERİ

- Sistem Yazılımı :
  - Diğer programlara hizmet sunmak üzere hazırlanmış programlar.
  - Derleyiciler, işletim sistemleri, vb.
- Mühendislik Yazılımı / Bilimsel Yazılım :
  - Mühendislik ve bilimsel hesaplamalarda kullanılmak üzere hazırlanmış programlar.
  - Büyük hacimli verilerle uğraşır.
  - "Numara öğretmek / Number crunching".
- Gömülü (Embedded) Yazılım :
  - Donanım ile çok sıkı ilişkidir.
  - Denetim amaçlıdır.
  - Gerçek zamanlı uygulamalardır.

5

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM TÜRLERİ

- Uygulama Yazılımı :
  - Product-line, shrink-wrapped, (commercial) off-the-shelf, vb.
    - Bkz. TS/BS ISO/IEC 25051 COTS Yazılım Ürünleri standardı
  - Bir çok mühendislik alanında olduğu gibi Yazılım Mühendisliği alanında da tanımlanmış standartlar vardır.
  - Erişim için kütüphaneye başvurunuz.
    - Ciddi bilgilere erişim için kütüphaneler kullanılmalıdır.
  - Farklı müşteriler tarafından kullanılabilecek genel amaçlı yazılımlar
  - Cari hesap uygulamaları, çeşitli otomasyon programları, kelime işlem uygulamaları, vb.
- Kurumsal Yazılım:
  - Belirli ticari iş gereksinimlerine yönelik programlar.
  - İş süreçleri ile ilgili bilgiye sahip olmalıdır.
  - Genellikle müşteriye özel tasarlanır.
  - Veri dönüştürme ve değerlendirme uygulamaları, iş süreçlerinin kimi zaman gerçek zamanlı izlenilmesi, vb.
  - Zamanla "eski yazılım" haline dönüşür!

6

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### ESKİ YAZILIM (Legacy Software):

- İş sürecinin önemli bir parçası olan ve çok uzun süredir kullanılan yazılımlar.
- Eski yazılımda bulunabilecek olumsuzluklar:
  - Eksik veya hatalı dokümantasyon
  - Zamanla karmaşıklaşmış kod
  - Esnek olmayan yapı
  - Eski donanım ile çok sıkı ilişki
  - Yazılım mühendisliğindeki gelişmelerden yoksunluk nedeniyle düşük kalite.
- Eski yazılımın değiştirilmesini gerektiren nedenler :
  - İş alanındaki yeni gereksinimler
  - Güncel sistemlerle birlikte çalışabilmesi için uyumluluk kazandırılması
  - Donanımın ömrünün dolması nedeniyle daha güncel ortama taşınma gerekliliği

7

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIMI ETKİLEYEN EĞİLİMLER

- Yaygınlaşan Bilgi-İşlem :
  - Hesaplama gücünün giderek küçülen alanlara sıkıştırılabilmesi, bilişimin günlük yaşantımızla daha kolay bütünleşmesine olanak sağlıyor.
- Yaygınlaşan Haberleşme Ağı :
  - Kablosuz ağların yaygınlaşması, bilişimin günlük yaşantımızla daha kolay bütünleşmesine olanak sağlıyor.
- Özgür / Açık Kaynak Yazılım :
  - Gevşek bir ekip tarafından geliştirilen yazılım, daha anlaşılır ve geliştirilebilir olmalıdır.
- Ayrıca:
  - Takım çalışması zorunluluğu
  - Küreselleşme
  - Ekonomik krizler

8

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM HAKKINDAKİ YANILGILAR: MÜŞTERİ AÇISINDAN

- Programın yazılmasına başlanması için amaçları genel olarak belirlemek yeter, ayrıntılar sonra kararlaştırılabilir. Nasıl olsa yazılım esnektir.
  - Belirsiz gereksinimler, çürük atılmış temele benzer.
- Yazılım esnektir. Değişen gereksinimler kolayca sisteme uyarlanabilir.
  - Yazılım yaşam döngüsünde ilerledikçe, değişen gereksinimleri yazılıma uyarlamanın bedeli üstel olarak artar.
- Sonuç: Yazılım esnek bir oyun hamurundan çok kil veya cam gibidir.
  - Çevik süreçlerle esnekliğin artırılması hedeflenmektedir.

9

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM HAKKINDAKİ YANILGILAR: PROGRAMCI AÇISINDAN

- Yazılımı tamamlayıp müşteriye teslim edince işimiz biter.
  - Yazılım üstünde harcanan çabanın yarısından fazlası, yazılımın müşteriye ilk teslimatından sonra harcanmaktadır.
- Yazılımı tamamlamadan kalitesini ölçmem.
  - Kalite güvence yöntemleri yazılım hayat döngüsünün her aşamasında uygulanabilir.
  - Çözümleme sürecinde dahi kullanılabilecek kalite ölçütleri bulunmaktadır.
- Yazılım eşittir program.
  - Gereksinim analizi başlı başına bir emektir.
  - Dokümantasyon ve sinama çalışmalarını da unutmayın!
  - Bazı durumlarda entegrasyon çalışmaları da gerekmektedir.
- Yazılım mühendisliğinin gereklerini uygulayarak boşuna çaba harcıyoruz.
  - Haritası olmayan yolunu kaybeder.
  - Kalite için harcanan çaba, karşılığını yazılım hayat döngüsünün ilerleyen aşamalarında fazlasıyla ödeyecektir.
  - Küresel ölçekte yazılım projelerinin %50'si başarısızlığa uğramaktadır.

10

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM HAKKINDAKİ YANILGILAR: İDARİ

- İşler yetişmiyorsa takıma yeni programcılar ekleriz.
  - Yazılım hayat döngüsü içerisinde ilerledikçe, yeni elemanların yazılıma hakim olması üstel olarak zorlaşır. İşler daha da gecikir.
- Geliştirmesini üstlendiğim yazılımı tamamen veya kısmen fason yaptırım.
  - Proje ilerlemesini kendi içinde denetleyemeyen bir firma, dışarıya verdiği işi izlemekte de zorlanacaktır.
- Açık kaynak yazılım üretirsem kar edemem.
  - Danışmanlık hizmetleri ile kar edilebilir.
  - Başka iş modelleri de vardır.

11

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme (Analysis)
- Tasarım (Design)
- Gerçekleme (Implementation)
- Sınama (Testing)
- Bakım (Maintenance)

### ÇÖZÜMLEME

- Çözümleme: Bir şeyi anlayabilmek için parçalarına ayırmak.
- Gerçeklenecek sistemi anlamaya yönelik çalışmalardan ve üst düzey planlama eylemlerinden oluşur.
  - Uygulama alanı
  - Kullanıcı gereksinimleri
  - Program parçaları arasındaki üst düzey ilişkiler ve etkileşimler (NYP'deki parçalar: sınıflar ve nesneler)
- "Bir sorunu anlamadan çözemezsiniz."

12

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme
- Tasarım
- Gerçekleme
- Sınama
- Bakım

#### TASARIM

- Tasarım: Bir araştırma ve/veya geliştirme sürecinin çeşitli dönemlerinde izlenecek yol ve işlemleri tasarlayan çerçeve.
- Çözümleme ile anlaşılan sorun tasarım aşamasında kağıt üzerinde (!) çözülür.
- Yazılım ↔ Tasarıma yönelik şemalar (NYP'de bazı tür UML şemaları), elektronik ↔ devre şemaları, mimari ↔ kat planları

#### GERÇEKLEME

- Eldeki tasarım, bir programlama dili ile kodlanır.

13

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme
- Tasarım
- Gerçekleme
- Sınama
- Bakım

#### SINAMA

- Sınama neden önemlidir?
  - Yazılım sürecinde ilerledikçe, ortaya çıkabilecek hataların giderilme maliyeti üstel olarak artar.
  - Aksi gibi, hataların büyük çoğunluğunun kökenleri isteklerin belirlenmesi ve tasarım aşamalarındaki sorunlara dayanır.
  - Bu yüzden: Erkenden, sık sık ve kolay sınama yapın.

14

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme
- Tasarım
- Gerçekleme
- Sınama
- Bakım

### BAKIM

- Yazılımın faaliyete geçirilmesinden sonra sistemde yapılan değişikliklerdir.
  - Yazılım hatalarının düzeltilmesi:
    - Kodlama hataları
    - Tasarım hataları (!)
    - Gereksinim ve analiz hataları (!!)
  - Sistemin işlevlerini değiştirme veya işlevlere eklemeler/çıkarmalar,
  - Yazılımın farklı bir ortama taşınması (programlama dili, işletim sistemi, donanım, iklim, vb.) (porting)

15

## YAZILIM MÜHENDİSLİĞİNE GİRİŞ

### YAZILIM SÜREÇLERİNİN GENEL ADIMLARI

- Çözümleme
- Tasarım
- Gerçekleme
- Sınama
- Bakım

### BAKIM

- Yeniden mühendislik (Refactoring / Software re-engineering)
  - Teknik bakış açısı: Yazılımın işlevini değiştirmeden iç yapısını değiştirmek.
  - Olası eylemler:
    - Yazılımın belgelendirilmesi
    - Tasarımın iyileştirilmesi/değiştirilmesi
    - Yazılımın farklı bir ortama taşınması

16



**YAZILIM MÜHENDİSLİĞİ DERS NOTLARI**  
**Yrd.Doç.Dr. Yunus Emre SELÇUK**

**YAZILIM GELİŞTİRME SÜREÇ (MODEL)LERİ**

17

**YAZILIM GELİŞTİRME SÜREÇLERİ**

- Yazılım geliştirme bir süreç olarak ele alınmalıdır.
  - Süreç: Önceden belirlenmiş adımlardan oluşan iş akışı.
- Süreç modelleri, yazılım geliştirme sürecinin yapısını ve adımlarını belirler.
  - Önceden ve iyi planlanmış bir süreç, zamanında ve 'kaliteli' bir 'ürün' elde edilmesini sağlar.
- Çeşitli modellerin kendine özgü avantaj ve dezavantajları vardır.
  - Gerçeklenecek projeye uygun modelin seçilmesi gerekir.

18

## YAZILIM GELİŞTİRME SÜREÇLERİ


### ŞELELE MODELİ

- Ardışıl Model / Şelale Modeli (Sequential / Waterfall)
  - Adımlar: Çözümleme → Tasarım → Kodlama → Sınama → Bakım.
  - Bir adımın tamamlanmasından sonra diğerine geçilir.
  - Eksiklikler veya hatalar fark edilirse bir önceki adıma geçilir.
- Artılar:
  - En eski model, yaygın kullanımda.
  - İyi tanımlanmış adımlar.
- Eksiler:
  - Son ürünün eldesi uzun süreceğinden müşteri sabırlı olmalıdır.
  - Adımları geride bıraktıkça, ilerleyen aşamalarda karşılaşılan hataların düzeltilmesi üstel olarak zorlaşmaktadır.
  - Bir çok 'müşteri' de gereksinimleri eksiksiz ve kesin belirtmekte zorlanmaktadır.
- Sonuç: Hiç model kullanmamaktan iyidir!
  - Önceden bir çok kez başarıya ulaştırılmış projelere benzer yeni projelerin yürütülmesi için kullanılabilir (rutin projeler).

19

## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÖN ÜRÜN MODELİ

- Ön ürün modeli / Prototip modeli
  - Adımlar: Müşteriyi dinle – Ön ürün oluştur – Müşteri ön ürünü dener –  

- Artılar :
  - Kullanıcı gereksinimlerinin daha iyi elde edilmesi.
  - Kullanıcının erkenden ürünü değerlendirmeye başlayabilmesi.
- Eksiler :
  - Ön ürün mükemmel değildir.
  - Eksik ürün zaman ve maliyet kısıtlamaları nedeniyle olgunlaşmadan canlı kullanıma alınabilmektedir.
- Sonuç: Prototip oluşturmayı başlı başına bir model olarak kullanmamalı, daha uygun bir modelin analiz aşamasında kullanılacak bir araç olarak ele almalı ve prototip ürünü silip atmalı.

20

## YAZILIM GELİŞTİRME SÜREÇLERİ

### HIZLI UYGULAMA GELİŞTİRME (RAD: Rapid Application Development)

- Kısa geliştirme çevrimleri üzerinde duran artımsal bir model.
- Ön koşullar:
  - Uygulamanın yaklaşık/ortalama 3 aylık bölümlere ayrılabilmesi,
  - Yeterli sayıda bölümün eşzamanlı ilerlemesinin sağlanabilmesi,
  - Yazılımın bileşenlerden kurulabilmesi.
- Artılar:
  - Bu sürece uygun yazılım projelerinde verimliliğin artması.
- Eksiler:
  - Büyük ölçekli çalışmalarda yeterli sayıda bölümü eşzamanlı ilerletebilecek sayıda çalışanın bulunamaması.
  - Çalışanlar hızla uyum sağlayabilmelidirler.
  - Yüksek teknik risklere uygun değil.
- Sonuç:
  - Prototip geliştirmede kullanılması veya ana fikirlerinin diğer süreçlere uygulanması yerinde olacaktır.

21

## YAZILIM GELİŞTİRME SÜREÇLERİ

### BİLEŞEN TABANLI (Component Based) UYGULAMA GELİŞTİRME

- Uygulamanın hazır yazılım bileşenlerinden oluşturulmasını öngörür.
- Aşamaları:
  - Konu alanı mühendisliği (Domain Engineering)
  - Aday bileşenlerin sınıflandırılması ve seçilmesi (Qualification)
  - Seçilen bileşenlerin kendi yazılımımıza uyarlanması (Adaptation)
  - Bileşenlerin bir araya getirilmesi (Composition)
- Artılar:
  - Yeniden kullanımın özendirilmesi (azalan giderler?)
- Eksiler:
  - Uygun bileşenlerin bulunması gerekliliği (her zaman bulunmaz)
  - Bileşenlerin uyarlanması gerekliliği (göründüğü kadar kolay olmayabilir)
- Sonuçlar:
  - Özellikle hızlı uygulama geliştirme olmak üzere, ana fikirleri çeşitli süreçlere uygulanabilir.

22

## YAZILIM GELİŞTİRME SÜREÇLERİ

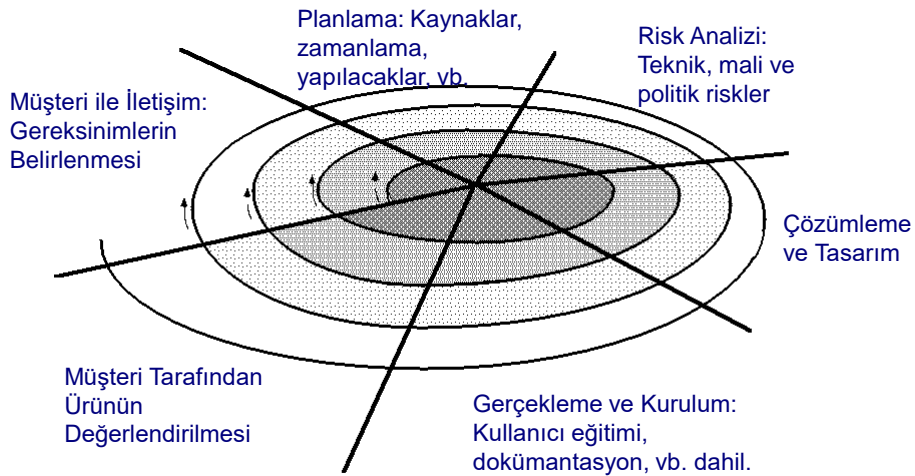
### ARTIMSAL / YİNELEMELİ MODELLER

- Artımsal / Yinelemeli Modeller (Incremental / Iterative)
  - Adımlar: Analiz – Tasarım – Kodlama – Sınama → Bakım
- Gereksinimler önemlerine ve birbirine bağımlılıklarına göre sıralanarak her yinelemede bunların bir kısmı tamamlanır.
- Artılar :
  - Ön ürün modeli ve ardışıl modelin güçlü yönlerini kendinde toplayarak dezavantajlarını geride bırakmıştır.
  - Nesneye yönelik programlama metodolojisi ile uyum içerisindedir.
- Eksiler : Yazılımın küçük artımlarına fazla yoğunlaşmak, sistemin geneline bakıldığında kolayca görülebilecek sorunların gözden kaçmasına neden olabilir.
- Sonuçlar:
  - Sistemin genelini göz ardı etmemek şartıyla güçlü bir modeldir.
- Örnekler: Spiral Model ve Kazan-Kazan Modeli

23

## YAZILIM GELİŞTİRME SÜREÇLERİ

### SARMAL (Spiral) MODEL



24

## KLASİK YİNELEMELİ SÜREÇLER

### Kazan-Kazan Modeli (WINWIN Model)



- Paydaş: Yazılımın başarısı ve başarısızlığının etkileyeceği kişi ve kurumlar.

25

## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÇEVİK (Agile) SÜREÇLER

- Değişen gereksinimler, teknik riskler gibi önceden belirlenemeyen durumlara ve yazılım ürününü etkileyebilecek her tür değişikliğe karşı esneklik sağlayan süreçlerdir.
- Bireyler ve etkileşimler
- Çalışan yazılım
- Müşterinin sürece katılımı
- Değişikliklere uyum sağlamak
- Süreçler ve gereçler
- Ayrıntılı belgeler
- Sözleşme pazarlığı
- Bir planı izlemek
- Çevik süreçler, sağ taraftaki maddelerin yararını kabul etmekle birlikte, sol taraftaki maddelere daha çok önem vermektedir.
- Bir ilerleme olmaksızın yalnızca sürekli uyum sağlamak başarı değildir.
  - Yazılımın artımsal gelişimi
  - Müşteriye erken ve sık ürün teslimi
  - Başarımın birincil ölçütü doğru çalışan yazılımdır.

26

## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÇEVİK (Agile) SÜREÇLER

- Çevik süreci yürütecek ekibin özellikleri:
  - Yüz yüze görüşme, en etkili bilgi aktarım yoludur.
  - Takım üyeleri çevik yaklaşım hakkında eğitilmelidir.
  - Ekip üyelerinin ortak amacı, çalışan yazılım üreterek müşteriye zamanında teslim etmek olmalıdır.
  - Ekip üyeleri birbirleriyle ve müşteriyle işbirliği içinde olmalıdır.
  - Ekip üyeleri karşılıklı saygı ve güven içerisinde olmalıdır.
  - Ekipler hem teknik, hem de tüm proje hakkında kararlar verebilmelidir.
  - Boşuna harcanan çaba yoktur: Çözülen bir sorun gereksizleşse bile, çözüm sürecinde edilen deneyim ekibe ileri aşamalarda yararlı olabilir.
- Kendi kendini düzenleme:
  - Ekibin kendisini yapılacak işe göre uyarlaması,
  - Ekibin kullanacağı süreci yerel ortama uyarlaması,
  - Üstünde çalışılan artımsal yazılım parçasını teslim etmek için gerekli çalışma zamanlamasını ekibin kendisinin belirlemesi.

27

## YAZILIM GELİŞTİRME SÜREÇLERİ


### ÇEVİK (Agile) SÜREÇLER

- Çevik süreçlerin dezavantajları:
  - Uygun olmayan ekiple çevik çalışılamamaktadır.
  - Kalabalık ekip veya büyük ölçekli projeler için uygun görülmemektedir.
  - Bir dış denetleyicinin dahil olduğu ve ayrıntılı kuralların gerektiği denetlemelerin zorunlu olduğu projelerde yetersiz kalmaktadır.
  - Çevik çalışmak disiplinsizlik olarak yorumlanmamalıdır.
- Çevik Süreç Örnekleri:
  - Aşırı Programlama (XP: Extreme Programming)
  - Scrum

28

## YAZILIM GELİŞTİRME SÜREÇLERİ


### ÇEVİK (Agile) SÜREÇLER

- Aşırı Programlama (XP)
    - Adımlar: Planlama – Tasarım – Kodlama – Sınama
- 
- Planlama:
    - Müşteri, kullanıcı öyküleri oluşturur.
      - Müşteri, öyküleri önemine göre derecelendirir.
      - Yaklaşık 3 haftada gerçekleşemeyecek öyküler varsa, ekip müşteriden bunları alt öykülere bölmesini ister.
    - Ekip ve kullanıcı, öykülerin sıradaki artımsal ürüne nasıl ekleneceğine karar verir:
      - Ya önce yüksek riskli öyküler gerçekleşir,
      - Ya da önce yüksek öncelikli öyküler gerçekleşir.
      - Her olasılıkta tüm öyküler kısa sürede (birkaç hafta) gerçekleşmelidir.

29

## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÇEVİK (Agile) SÜREÇLER

- Aşırı Programlama (XP)
    - Adımlar: Planlama – Tasarım – Kodlama – Sınama
- 
- Planlama (Devam):
    - İlk artımsal ürün projenin hızını ölçme amacıyla değerlendirilir:
      - Eldeki artımın hızına göre sonraki artımların teslim tarihleri belirlenir.
      - Aşırı sözler verildiği ortaya çıkarsa artımsal ürünlerin içeriği de yeniden kararlaştırılabilir.
    - Süreç ilerledikçe müşteri yeni öyküler ekleyebilir, eski öykülerin önceliğini değiştirebilir, öyküleri farklı şekillerde bölüp birleştirebilir, bazı öykülerden vazgeçebilir.
      - Bu durumda ekip kalan artımları ve iş planlarını uygun biçimde değiştirir.

30

## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÇEVİK (Agile) SÜREÇLER

- Aşırı Programlama (XP)
  - Adımlar: Planlama – Tasarım – Kodlama – Sınama → Artımsal Ürün
- Tasarım:
  - Basit tasarım karmaşık gösterimden üstündür. (KISS: Keep It Simple, Stupid!)
  - CRC (Class-Responsibility-Collaboration) kartları ile yazılımın sınıf düzeyinde incelenmesi.
  - Karmaşık bir tasarımdan kaçınılamazsa işlevsel bir ön gerçekleştirme yapılır (Spike solution).
  - Refactoring teşvik edilir.
  - Bu aşamanın ürünleri CRC kartları ve ön gerçeklemlerdir (başka ürün yok).

31

## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÇEVİK (Agile) SÜREÇLER

- Örnek CRC kartı:

Sınıf adları


<b>Sınıf: Satış</b>	
Kasada yapılan ödemeyi simgeleyen sınıf.	
<b>Üst Sınıf(lar):</b> Yok	
<b>Alt Sınıf(lar):</b> Yok	
<b>Sorumluluk:</b>	<b>İşbirlikçi:</b>
Satışın yapıldığı tarih ve saati saklamak	
Yapılan ödeme tutarını saklamak	Ödeme
Satılan malların listesine erişim	Mal

32



## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÇEVİK (Agile) SÜREÇLER


- Aşırı Programlama (XP)
  - Adımlar: Planlama – Tasarım – Kodlama – Sınama

Artımsal Ürün
- Kodlama:
  - Önce birim sınamaları hazırlanır.
    - Programcı tarafından yapılan, sınıfların (NYP'de; yapısal'da fonksiyonlar, vb.'lerin) temel işlevselliklerini sınama amaçlı kod.
    - Sadece sınavı geçmeye yarayan kod yazılır (KISS).
  - Çift kişi ile kodlama:
    - Bir programcı eldeki sorunu çözerken diğeri çözümün genel tasarıma uygunluğunu gözetir ve kodlamanın takımın karar verdiği ölçütlere (kalite, vb.) uygunluğunu denetler.
- Sınama:
  - Birim sınamalarının otomatik çalıştırılması.
  - Müşterinin artımsal ürünü denemesi.

33

## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÇEVİK (Agile) SÜREÇLER

- Scrum:
  - Adımlar: Görev Listesi – Koşu – İşlev Gösterimi
- Görev Listesi = Kullanıcı öyküleri.
  - Önceliklendirilmiştir.
- Koşu:
  - Görev listesinin maddelerinden biri seçilir ve önceden belirlenmiş kısa bir süre içerisinde (Ör. 1-4 hafta) gerçekleşir.
  - Koşu süresince ekibin her gün yaptığı kısa (Ör. 15dk) toplantılar:
    - Proje lideri yönetir.
    - Cevaplanmaya çalışılan üç ana soru:
      - Son toplantıdan bu yana ne yaptınız?
      - Karşılaştığınız engeller nelerdir?
      - Yarınki toplantıda neleri başarmayı hedefliyorsunuz?
- İşlev Gösterimi: Müşterinin en yeni işlevi veya o ana dek gerçekleştirilen tüm işlevleri sinaması.

34

## YAZILIM GELİŞTİRME SÜREÇLERİ

### ÇEVİK (Agile) SÜREÇLER

- Çevik Modelleme
  - Bir amaç için modelleme yapın:
    - Neyi, kime, hangi düzeyde anlatmak istiyorsunuz?
    - Buna göre uygun modelin ve ayrıntılandırmanın seçimi .
  - İçerik sunumdan daha önemlidir.
    - Gerekli bilgiyi içermeyen hatasız model işe yaramaz!
  - Kullandığınız modelleme yolunun özünü ve modellerinizi oluşturmak için kullanacağınız araçları iyi öğrenin.
- DİKKAT: Önemli olan dengeyi korumaktır.
  - Çevik çalışacağız diye serseri programcı olmayın.
  - Disiplinli çalışacağız diye sırtınızda tuğla çuvalı taşımayın.

35

## YAZILIM GELİŞTİRME SÜREÇLERİ

### SÜREÇ SERTİFİKASYONU

- Olgunlaşmış bir yazılım geliştirme sürecine sahip olmayan bir yazılım firması, projelerini başarı ile sonuçlandıramaz.
- Bir yazılım firması, süreçlerinin yeterliliğini bağımsız kurumlara onaylatmayı seçebilir.
- Gerekli olduğu durumlar:
  - Bazı büyük müşteriler sertifikalı yazılım firmaları ile çalışmayı şart koşarlar.
- Gereksiz olduğu durumlar:
  - Çok küçük şirketler ve/veya projeler için ek yük olarak görülebilir.
- Güncel model ve standartlar:
  - CMMI: Capability Maturity Model Integration
    - SEI tarafından önerilmiştir (Software Engineering Institute of Carnegie-Mellon University)
  - PMI: Genel amaçlı bir proje yönetimi yaklaşımı
  - ISO 9001:2000 standartları (Genel)
    - ISO/IEC 90003:2004 (Yazılım geliştirmeye özel)
  - Genel vs. Özel (Peynir mi üretiyoruz?)

36

## YAZILIM GELİŞTİRME SÜREÇLERİ

### CMMI DÜZEYLERİ

- CMMI düzeyleri:
  - 1. Düzey: Giriş düzeyi (Level 1: Initial). İş şansa ve anahtar kişilere kalmış.
  - 2. Düzey: Yinelenebilir (Repeatable). Temel planlama ve izleme yöntemleri kullanılarak, önceki projelerdeki başarılar yeni projelerde tekrarlanılabilir.
  - 3. Düzey: Tanımlanmış (Defined). Kişi ve risk yönetimi ile projenin yönetimi iyileştirilir.
    - Büyük müşteriler en azından bu düzeyde yazılım evleri ile çalışmak ister.
  - 4. Düzey: Yönetilen (Managed). Süreç ve yazılım ölçütleri kullanılarak kalite yönetimine geçilir. İlerleme sürekli izlenir, bütçe ve zaman hedeflerinden sapmalar erkenden belirlenerek gerekli önlemler alınır.
  - 5. Düzey: İyileştirilmiş (Optimized). Süreç yönetimi geçmiş deneyimlerin ışığında sürekli iyileştirilir.
- 700'den fazla sayfaya sahip dokümanı için: <http://www.sei.cmu.edu/cmmi/>

37

## YAZILIM GELİŞTİRME SÜREÇLERİ

### CMMI DÜZEYLERİ

- CMMI, her düzeyde belli süreç alanlarının kapsanıyor olmasını ister.
  - Süreç alanları belli hedeflere ulaşmak için beklenen uygulamalardır.
  - Her firma gerekli süreç alanlarını kendine özgü süreçlerle kapsar.
- CMMI türleri:
  - CMMI-DEV (Development): Yazılım geliştirme
  - CMMI-SVC (Service): Hizmet sunumu ve yönetimi
  - CMMI-ACQ (Acquisition): Ürün ve hizmet alımı
- CMMI Level 3+ sertifikası almış kamu ve özel kurumlarımıza örnekler:
  - MilSoft (Level 5)
  - TÜBİTAK BİLGEM Yazılım Teknolojileri Araştırma Enstitüsü (Level 4)
  - ASELSAN (Level 3)
  - Cybersoft (Level 3)
  - Havelsan (Level 3)
  - Koç Sistem (Level 3)
  - Ayrıntılar: <https://sas.cmmiinstitute.com/pars/pars.aspx>

38

## YAZILIM MÜHENDİSLİĞİ DERS NOTLARI

Yrd.Doç.Dr. Yunus Emre SELÇUK

### GEREKSİNİM MÜHENDİSLİĞİ

39

## GEREKSİNİM MÜHENDİSLİĞİNE GİRİŞ

### GEREKSİNİM MÜHENDİSLİĞİ

- Üzerinde çalışılmaya başlanacak projenin amaçlarını, boyutlarını ve etkilerini belirlemeye yönelik çalışmalardır.
  - Genel amaçlı proje yönetimi faaliyetleri arasında yer alan yapılabirlik (feasibility) çalışmasına bir girdi olarak düşünülebilir.
- Müşteri ne istediğini bilmez mi? Gereksinimler zaten belli değil mi?
  - Çoğunlukla müşterinin kafasında sadece genel bir fikir vardır.
  - Yoruma açık ve ayrıntıları kesin çizgilerle belirlenmemiş gereksinimler projenin başarısızlığına davetiye çıkarır.
  - Kesin belirlenmiş gereksinimler bile zaman içerisinde değişebilir.
- Değişler:
  - Şeytan ayrıntıda gizlidir.
  - Yanlış veya eksik işi yapan mükemmel yazılım değil, doğru işi yapan iyi çözüm gereklidir.
  - SONUÇ: Gereksinim mühendisliği gerekli bir etkinliktir.

40

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Gereksinim mühendisliğinin genel adımları:
  - Başlangıç (Inception)
  - Bilgi Toplama (Elicitation)
  - İşleme (Elaboration)
  - Pazarlık (Negotiation)
  - Tanımlama (Specification)
  - Doğrulama (Validation)
  - Yönetim (Management)
- Gereksinim mühendisliği adımları gerçekleştirilecek yazılımın doğasına ve kullanılan sürece göre düzenlenmelidir.
- Gereksinim mühendisliği adımları süresince yazılım ekibi ve müşteri birlikte çalışmalıdır.
  - Müşterinin bir ekibinin, yazılım geliştirme sürecinin mümkün olduğunca çok adımının bir parçası olması yararlıdır.

41

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
  - Bilgi Toplama (Elicitation)
  - İşleme (Elaboration)
  - Pazarlık (Negotiation)
  - Tanımlama (Specification)
  - Doğrulama (Validation)
  - Yönetim (Management)
- Başlangıç:
    - Yazılım projesinin **ilk aşamalarının** başlatılıp başlatılmamasına karar verilen adımdır.
    - Müşterinin bir yazılım projesi başlatılmasını düşünmesine neden olan olaylar:
      - Yeni bir iş gereksiniminin belirlenmesi.
      - Mevcut iş süreçlerinde güçlüklerle karşılaşılması.
      - Müşterinin üst düzey karar vericileri ve astları arasında geçen kısa bir sözlü konuşma veya toplantı ile bile bir proje başlayabilir.
    - Bir uygulama yazılımı söz konusu ise:
      - Yeni bir pazarın veya hizmetin farkına varılması,
      - Yazılım şirketinin üst düzey karar vericileri ve teknik ekibinin sözlü konuşması ile yeni bir yazılım projesi başlatılabilir.

42

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)
- Başlangıç aşamasında paydaşlar belirlenmelidir.
  - Paydaş: Gerçeklenecek sistemden doğrudan veya dolaylı olarak yararlanabilecek ve etkilenebilecek herkes.
    - Her paydaş sisteme farklı bir açıdan bakar.
    - Projenin başarısı veya başarısızlığı paydaşları farklı şekillerde etkiler.
    - Paydaşlara sorulacak sorularla belirlenmesi gerekenler:
      - Paydaşların bakış açıları,
      - Paydaşları etkileyebilecek nedenler,
      - Söz konusu etkilerin sonuçları.

43

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)
- Bilgi toplama aşamasının genel ilkeleri:
  - Gereksinimler hakkında ayrıntılı bilgiler, tüm paydaşların etkin katılımı ile elde edilmelidir.
  - Tüm paydaşların katıldığı toplantılar yapılmalıdır.
  - Toplantılara hazırlık ve katılım kuralları belirlenmelidir.
  - Gündem belirlenmelidir: Önemli konuları atlamayacak kadar sıkı, yaratıcılığı önlemeyecek kadar açık olmalıdır.
  - Düzeni sağlayacak ve tikanıklıkları çözecek bir oturum başkanı seçilir.

44

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)
- İşleme:
  - Bilgi toplama aşamasında toplanan 'ham' bilgilerin 'işlenmesi'.
  - Son kullanıcının ve diğer paydaşların yazılımla nasıl etkileşimde bulunacağını belirlenmesi ve ayrıntılandırılmasını amaçlar.
  - Etkileşimler, kullanım senaryoları ile gösterilir (ileride anlatılacak).
  - İşleme kimi bilgilerin genişletilmesi, kimi bilgilerin özetlenmesi şeklinde gerçekleşir.
  - Gereksinimlerin sınıflandırılması
    - Normal gereksinimler
    - Beklenen gereksinimler: Çok temel gereksinimleri kullanıcı belirtmeyebilir. Bunların da elde edilmesi gereklidir.
    - Heveslendirici gereksinimler: Müşteri beklentilerinin ötesinde ve varlığında müşteriyi sevindirecek özellikler.

45

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)
- Pazarlık:
  - Müşteriler sınırlı insan, zaman ve bütçe kaynakları çerçevesinde karşılanamayacak aşırı isteklerde bulunabilir.
  - Paydaşlar gereksinimleri farklı önem düzeylerinde görebilir.
  - Farklı paydaşların gereksinimleri birbiri ile çelişebilir.
  - Pazarlık sonucunda tüm paydaşların razı olacağı bir gereksinimler listesi elde edilir.

46

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)
- Tanımlama:
  - Gereksinimler tanımlama aşamasında, pazarlık sonucu üzerinde uzlaşılan haliyle kağıda dökülür.
  - Tanımlama araçları:
    - Konuşma dili ile yazılmış belgeler
    - Kullanıcı senaryoları: Görülecek
    - Kullanım şemaları: Görülecek
    - Formel modeller (Matematiksel gösterim, işlenilmeyecek)
    - Bir ön ürün
  - Birden fazla tanımlama aracı birlikte kullanılabilir.

47

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)
- Doğrulama:
  - Tanımlanmış gereksinimlerin tutarsızlıklara karşı sağlanması yapılır.
    - Gereksinimler açıkça ve yoruma yer bırakmayacak şekilde tanımlanmış mı?
    - Birbiri ile çelişen gereksinimler var mı?
    - Gereksinimlerde hatalar ve eksikler var mı?
    - Eksik gereksinimler var mı?
    - Gerçekçi olmayan gereksinimler var mı?
    - ...
  - Doğrulama yapma için önerilen temel yol teknik değerlendirmedir (Formal technical review, sına teknikleri arasında anlatılacak).

48



## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)
- Yönetim:
  - Yazılım geliştirme süreci içerisinde gereksinimlerde değişiklikler olabilir:
    - Yeni gereksinimler eklenmesi
    - Mevcut gereksinimlerden bazılarının geçerliliğini yitirmesi
    - Gereksinimlerin önem sıralamasının değişmesi
    - Hatalı kestirimlerden dolayı bazı gereksinimlerden vazgeçilmesi
  - Gereksinimlerde ne tür değişikliklerin nasıl ve hangi şartlarla yapılabileceği, resmi bir sözleşme ile önceden belirlenebilir.
  - Gereksinimlerde değişiklikler müşteri ile karşılıklı anlaşma ile yapılmalıdır.

49

## GEREKSİNİM MÜHENDİSLİĞİ

### GEREKSİNİM MÜHENDİSLİĞİ ADIMLARI

- Başlangıç (Inception)
- Bilgi Toplama (Elicitation)
- İşleme (Elaboration)
- Pazarlık (Negotiation)
- Tanımlama (Specification)
- Doğrulama (Validation)
- Yönetim (Management)
- Yönetim (devam):
  - Yazılım geliştirme süreci içerisinde gereksinimlerin gerçekleşmesinin (ve varsa gereksinimlerdeki değişikliklerin) izlenmesi gerekir.
  - İzleme tablolar aracılığı ile yapılır:
    - İzlenebilirlik tabloları (Traceability table).

	B1	B2	B3	...
G1	✓		✓	
G2		✓		
G3	✓			
...				

- G1,2,...: Gereksinimler
- B1,2,...: Sisteme çeşitli bakış açıları
  - Modüller, Paketler, Sınıflar, vb.

50

**YAZILIM MÜHENDİSLİĞİ TEMELLERİ DERS NOTLARI**  
**Yrd.Doç.Dr. Yunus Emre SELÇUK**

**NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ**

51

**NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ**

**NESNEYE YÖNELİK ÇÖZÜMLEMENİN TEMELLERİ**

- Çözümleme (Analiz): Bir şeyi anlayabilmek için parçalarına ayırmak.
- Sistemi anlamaya yönelik çalışmalardan ve üst düzey planlama eylemlerinden oluşur.
  - Uygulama/problem alanının anlaşılması.
  - Kullanıcı gereksinimlerinin anlaşılması.
  - Koddaki sınıflar ve nesneler ile bunların arasındaki üst düzey etkileşimlerin belirlenmesi: Çözümleme modelinin oluşturulması.
- “Bir sorunu anlamadan çözemezsiniz.”

52

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### UYGULAMA ALANININ ÇÖZÜMLENMESİ (DOMAIN ANALYSIS)

- Amaç, uygulama alanını anlamak ve elde edilen bilgileri analiz modeline taşımaktır.
- Uygulama alanı hakkında bilgi edinilebilecek kaynaklar:
  - Teknik literatür
  - Mevcut uygulamalar
  - Müşteri anketleri
  - Uzman tavsiyeleri
  - Mevcut ve gelecekteki gereksinimler
- Problem alanı hakkında bilgi edinmeden “müşterinin dilinden konuşamazsınız”.

53

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### GEREKSİNİMLERİN BELİRLENMESİ

- Gereksinimler belgesi:
  - Müşterinin programdan beklentilerini anlatan, doğal konuşma dili ile yazılmış belge.
- Örnek gereksinimler belgesi:

#### **NextGenPOS Perakende Satış Programı**

Eski yazılım ihtiyaçlarımızı karşılayamadığından, yenilenecek donanımla birlikte perakende satış programımızın da yenilenmesine gerek duyuyoruz. Program kasada yapılan alış-veriş işlemlerine yardımcı olmalıdır. Yapılan her işlem program tarafından saklanmalı; mali bilgiler harici bütçe sistemine, mal çıkış bilgileri ise harici envanter sistemine iletilmelidir. Saklanan işlemler üzerinde daha sonra raporlamalar ve analizler yapılabilir. Sistem yapılan alış-verişler karşılığında müşteriye fiş vermelidir. Yapılan her satış için KDV de hesaplanarak ayrıca belirtilmelidir. Şirketimizin birden fazla şubesi olup tüm şubelerdeki işlemler merkezi sunucuya iletilmelidir.

- Doğal dille yazılmış gereksinimler belgesinden kullanım öykülerine geçiş yapılır.

54

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### GEREKSİNİMLERİN BELİRLENMESİ

- Kullanım öyküleri:
  - Programın yapacağı işleri ayrıntılı adımlarla ve belli kurallara uyarak anlatan belgeler.
- Kullanım öykülerinin oluşturulmasındaki amaç:
  - Ürünün sağlaması beklenen işlevleri ve ürünün çalışma ortamını belirlemek,
  - Son kullanıcı ve yazılım ekibi arasında bir anlaşma zemini belirlemek,
  - Son kullanıcı ve sistemin birbirleri ile nasıl etkileşimde bulunacağını açık ve belirsizlikten uzak olarak tanımlamak,
  - Doğrulama testleri için bir zemin oluşturmak.
- Bir kullanım öyküsünün bölümleri:
  - Giriş bölümü: Sistemin neyi hangi koşullar ve sınırlar içerisinde yapması gerektiğini anlatır.
  - Ana senaryo / Ana başarılı akış: Her şeyin yolunda gitmesi halinde yürütülecek eylemler.
  - Alternatif senaryolar: Bir aksilik olması halinde yapılacak işlemler.

55

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### KULLANIM ÖYKÜSÜ: Satış İşlemi

**Birincil Aktör:** Kasiyer.

**İlgililer ve İlgili Alanları:**

- Kasiyer: Doğru ve hızlı giriş ister, kasa açığı maaşından kesildiğinden ödeme hataları istemez
- Satıcı: Satış komisyonlarının güncellenmesini ister
- Müşteri: En az çaba ile hızlı hizmet ister. Vergi iadesinde veya mal iadesinde kullanmak üzere fiş ister.
- ...

**Ön Koşullar:**

- Kasiyerin kimliği doğrulanır.

**Son Koşullar:**

- Satış kaydedilir. KDV doğru olarak hesaplanır. Mali kayıtlar ve envanter kayıtları güncellenir. Komisyonlar kayıt edilir. Fiş yazılır. Ödeme emri onayları kaydedilir.

- Dikkat: Kullanım öyküsünde yer alacak her şey, verilen ilgi alanlarına giren şeyler olmalıdır.
- Aktör: Sistem ile etkileşimde bulunan varlıklar.
  - İnsan
  - Yazılım veya donanım.

56

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### KULLANIM ÖYKÜSÜ: Satış İşlemi

#### Ana Öykü:

1. Müşteri kasaya alacağı mallarla gelir.
2. Kasiyer yeni bir satış işlemi başlatır.
3. Kasiyer ürünün barkodunu girer.
4. Sistem bir satış kanalı maddesi oluşturur. Bu maddede ürün tanımı, fiyatı ve toplam bedel (aynı maldan birden fazla alınmış olabilir) yer alır.
5. Kasiyer 3. ve 4. adımları müşterinin alacağı tüm mallar için tekrarlar.
6. Sistem toplam bedeli vergi iadesi ile birlikte hesaplar.
7. Kasiyer müşteriye toplamı bildirir ve ödeme ister.
8. Müşteri ödemeyi yapar ve sistem ödemeyi tahsil eder.
9. Sistem tamamlanan işlemin kaydını tutmayı tamamlar ve harici envanter ile mali sistemlere gerekli bilgileri gönderir.
10. Sistem makbuz verir.
11. Müşteri mallarla birlikte ayrılır.

57

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### KULLANIM ÖYKÜSÜ: Satış İşlemi

#### Alternatif Öyküler:

##### 3a. Geçersiz barkod

1. Sistem uyarı mesajı verir ve kayıt girişini reddeder.

##### 3-7a. Müşteri bir kalem mali alışverişten çıkartmak ister.

1. Kasiyer satıştan çıkarmak üzere ürünün barkodunu okutur.
2. Sistem güncel toplamı bildirir.

...

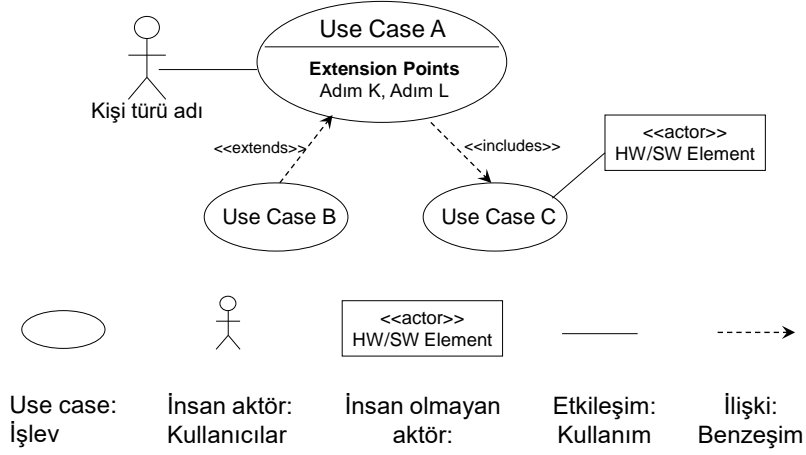
### KULLANIM ÖYKÜLERİNİN GRAFİK GÖSTERİMİ

- Kullanım öyküleri, ayrıntılı ve uzun belgelerdir.
- Yazılımın yapacağı işlerin özet gösterimi için kullanım şemaları çizilir (use-case diagrams).
- Çizim kurallarını verdikten sonra örnek öykünün şemasını çiz.

58

## KULLANIM ŞEMALARI – USE CASE SCHEMAS

### ÇİZİM KURALLARI



59

## KULLANIM ŞEMALARI – USE CASE SCHEMAS

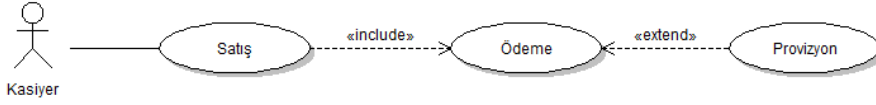
### ÇİZİM BİLGİLERİ

- Benzeşim ilişkileri:
  - Ok yönü aynı zamanda ilişkiyi okuma yönüdür.
  - UC-B extends UC-A : B işlevi, A işlevi yürütülürken oluşabilecek bir sapış anlamındadır.
    - A: Ana akış
    - B: Ana akıştaki bir seçenek, ana akıştan bir sapış, alt akış
  - UC-A includes UC-C: A işlevi, C işlevini içerir.
    - A: Ana akış, içeren akış
    - C: Alt akış, içerilen akış

60

## KULLANIM ŞEMALARI – USE CASE SCHEMAS

### ÖRNEK ÇİZİM



- Bir POS yazılımının ödeme işlevini kasiyer kullanır.
- Satış işlevi, içerisinde ödeme yapma işlevini içerir.
  - Includes, çünkü: Her satış içerisinde mutlaka ödeme olur.
- Ödemenin kredi kartı ile olması halinde, provizyon alma işlemi yürütülür.
  - Extends, çünkü: Ödeme nakit ise provizyona gerek kalmaz.
  - Provizyon: Kredi kartının limitinin aşılp aşılmadığı, çalıntı olup olmadığı, vb. gibi bilgilerin sınanması anlamında bir bankacılık terimi.

61

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### SINIFLARIN BELİRLENMESİ

- Kullanıcı gereksinimleri belgesinden ve kullanım senaryolarından sınıfların elde edilmesi.
  - İsimlerin taranarak aday sınıfların elde edilmesi.
  - Adaylar aşağıdaki kurallardan birini sağlamalıdır:
    1. Saklanan bilgi: Sistemin çalışması süresince bu varlığın durumu saklanmalıdır.
    2. Gereksinim duyulan hizmetler: Bu varlığın hizmetlerine ihtiyaç duyan başka varlıklar vardır.
    3. Gerekli varlıklar: Problemin çözümü ile ilgili bilgi üreten veya problemin çözümü için bilgi tüketen varlıklar.
  - Değınilen kurallardan birini sağlayamayan adayları, bir başka sınıfın üye alanı olarak değıerlendirebiliriz.
- Örnek gereksinim belgesinden sınıfları oluşturun.

62

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### SINIFLARIN BELİRLENMESİ

- Üyelerin belirlenmesi:
  - Sıfat ve eylemlerin taranması
  - Sorumlulukların belirlenmesi (CRC kartları)
- Sorumlulukların dağıtılması:
  - Sorumlulukların bir yerde yoğunlaşmaması
  - Sorumlulukların genelden özele doğru tanımlanması (kalıtım hiyerarşisinde genelden özele gidilmesi)
  - Bir bilgi ile ilgili davranışların, o bilgi ile aynı sınıfta yer alması (encapsulation)
  - Tek bir şey hakkındaki bilginin tek sınıfta yer alması
  - Gerekli sorumlulukların paylaşılması

63

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### ETKİLEŞİMLERİN BELİRLENMESİ

- Etkileşim: Bir nesnenin üzerine düşen sorumluluğu yerine getirmek için diğer bir nesneye mesaj göndermesi.
- Nesneler arasındaki ilişkiler
  - Bağlantı, toplama, meydana gelme.
- Sınıflar arasındaki ilişkiler
  - Özelleşme/genelleşme
- Çözümleme aşamasında ne tür etkileşimlerin olabileceği düşünülür, etkileşimlerin nasıl olacağı düşünülmez.
- Bu konuların temeli "Nesneye Dayalı Kavramlar ve Programlama" dersinde atılmıştır.
- İlerleyen yansılarda UML sınıf şemalarının temel kuralları hatırlatılacaktır.

64



## NESNEYE YÖNELİK PROGRAMLAMANIN TEMELLERİ

### NESNELER VE SINIFLAR

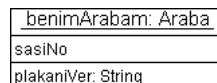
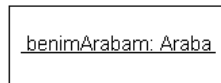
- Nesne: Durum bilgisi ve tanımlı eylemler içeren, temel programlama birimi.
  - Nesne = varlık.
  - Nesnenin nitelikleri = Nesne ile ilgili veriler.
  - Eylemler = Nesnenin kendi verisi üzerinde(\*) yapılan işlemler = Nesnenin kendi verileri ile çalışan metotlar (fonksiyonlar).
    - \*Çeşitli kurallar çerçevesinde aksi belirtilmedikçe.
- Sınıf: Nesneleri tanımlayan şablonlar.
  - Sınıf = tür / tip
- Örnek nesne: Bir otomobil.
  - Nitelikler: Modeli, plaka numarası, rengi, vb.
  - Eylemler: Hareket etmek, plaka numarasını öğrenmek, satmak, vb.
- Örnek sınıf: Taşıt aracı.
  - Nitelikleri ve eylemleri tanımlayan program kodu.
- Bir nesneye yönelik program içerisinde, istenildiği zaman herhangi bir sınıftan olan bir nesne oluşturulabilir.
- Aynı anda aynı sınıftan birden fazla nesne etkin olabilir.

65

## SINIFLAR, NESNELER VE MESAJLAR

### TERMİNOLOJİ VE GÖSTERİM

- NYP Terminolojisi:
  - Veri: Üye alan (Member field) = Nitelik (attribute)
  - Durum bilgisi: Nesnenin belli bir andaki niteliklerinin durumları
  - Eylem: Metot (Member method)
  - Nesnenin (Sınıfın) üyeleri = Üye alanlar + üye metotlar
  - Sınıf = tür = tip.
  - S sınıfından oluşturulan n nesnesi = n nesnesi S sınıfının bir örneğidir (instance)
- UML Gösterimi (1.1):



66

## SINIFLAR, NESNELER VE MESAJLAR

### HER NESNE FARKLIDIR!

- Aynı türden nesneler bile birbirinden farklıdır:
  - Aynı tür niteliklere sahip olsalar da, söz konusu nesnelerin nitelikleri birbirinden farklıdır = Durum bilgileri birbirinden farklıdır.
  - Durum bilgileri aynı olsa bile, bilgisayarın belleğinde bu iki nesne farklı nesneler olarak ele alınacaktır.
  - Örnek: Sokaktaki arabalar.
    - Modelleri ve renkleri farklı olacaktır.
    - Aynı renk ve model iki araba görseniz bile, plakaları farklı olacaktır.
    - Plaka sahteciliği sonucu aynı plakaya sahip olsalar bile, bu iki araba birbirlerinden farklı varlıklardır.

67

## NESNEYE YÖNELİK PROGRAMLAMANIN TEMELLERİ

### NESNELER VE SINIFLAR

- Veri gizleme (Data/Information Hiding)
  - Üye alanlara üye metotlar ile erişim (getter, setter methods)
  - İç mekanizmayı gizler (kara kutu yaklaşımı/soyutlama)
  - Yeniden kullanılabilirliği artırır
- Erişim kısıtlamaları
  - public, private
  - package / protected
- Sınıfa erişim ne kadar kısıtlanmalı?
  - Kendisinden beklenen hizmetleri sağlayabilecek kadar açık,
  - Kullanıcı gereksiz ayrıntılarla uğraştırmayacak ve sınıf bütünlüğü ile uygulama güvenliğini tehlikeye atmayacak kadar kapalı sınıflar.

<<Interface/abstract/vb.>>  
Sınıf Adı

+ public üye: Tip  
- private üye: Tip  
# protected üye: Tip  
~ package üye: Tip

+ metod adı( Param.1 tipi,  
P.2 tipi, ... ): Dönüş Tipi

68

## NESNEYE YÖNELİK PROGRAMLAMANNIN TEMELLERİ

### NESNELER VE SINIFLAR

- Nesneler arasındaki ilişkiler (relations):
  - Bağlantı (Association)
  - Kullanma (Dependency)
  - Toplama (Aggregation)
  - Meydana Gelme (Composition)
  - Kural Koyma (Associative)
  - Kalıtım (Inheritance)

#### Bağlantı (Association)

- Bir nesnenin diğerinin yeteneklerini kullanmasıdır.
- En basit ilişki şeklidir.
- Sadece ilişki kelimesi geçiyorsa, ilişkinin iki nesne arasındaki bağlantı olduğu anlaşılır.
- SOR: Bir nesnenin diğerinin yeteneklerini kullanması nasıl olur?
  - Yanıt: Görülebilirlik kuralları çerçevesinde ve metotlar üzerinden.
  - Yani: Mesaj göndererek.

69

## NESNELER ARASINDAKİ İLİŞKİLER

### Bağlantı (Association)

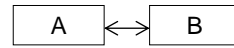
- Gösterim:



Bağımlılık

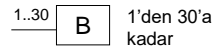
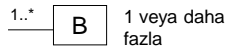
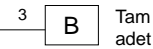
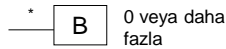


A, B'ye bağımlı: b nesneleri a nesnelerinden habersiz.



Çift yönlü bağımlılık

- Ok yoksa:
  - Ya çift yönlü bağımlılık vardır,
  - Ya da henüz bağımlılığın yönü düşünülmemiştir.
- İlişkinin uçlarında sayılar olabilir (cardinality)
  - Çoğulluk ifade eder.
  - İlişkinin o ucunda bulunan nesne sayısını gösterir.



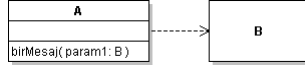
- SOR: Bir çelişki yok mu?
  - Yanıt: Nesneler arasındaki ilişkiler dedik ama sınıf gösteriliyor.
  - Kasıtlı değil, gösterim öyle.

70

## NESNELER ARASINDAKİ İLİŞKİLER

### KULLANMA İLİŞKİSİ (DEPENDENCY)

- Bir nesne diğerine giden bir mesajın parametresi ise.
- Burada anahtar kelime **parametre olmaktır**.
- Gösterim:



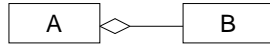
A, B'yi kullanır: b nesneleri a nesnelerinden habersiz.

71

## NESNELER ARASINDAKİ İLİŞKİLER

### Toplama (Aggregation)

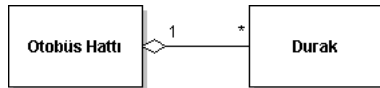
- Parça-bütün ilişkisini simgeler.
- Gösterim:



Toplama (aggregation)

- A örneği B örneklerinin birleşmesinden oluşur.
- A: Bütün, B: Parça.

- Bir nesnenin diğer nesnelerin birleşmesinden oluştuğunu anlatır.
- Fiziksel veya mantıksal birleşme söz konusu olabilir.
- Zayıf bir birlikteliği simgeler. Şu gibi konular dikkate alınmaz:
  - Parçaların tek başlarına anlamlarının olup olmadığı
  - Parçaları oluşturma ve birleştirme sorumluluğu
  - Parçasız bir bütünün var olup olmayacağı

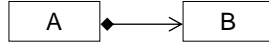


72

## NESNELER ARASINDAKİ İLİŞKİLER

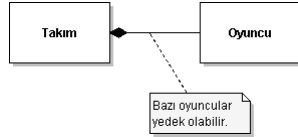
### Meydana Gelme (Composition)

- Parça-bütün ilişkisini simgeler.
- Farkı: Daha kuvvetli bir birlikteliği anlatır.



Meydana gelme  
(composition)

- Birlikteliğin kuralları:
  - Parçaların tek başlarına anlamları yoktur.
  - Parçaları oluşturma ve birleştirme, bütünü simgeleyen nesnenin sorumluluğundadır.
  - Diğer kurallar UML şemasındaki notlarla veya kod ile ilgili başka tür belgelerle ayrıntılandırılır.

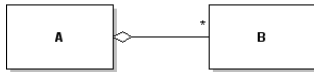
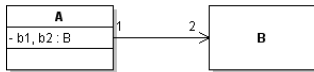
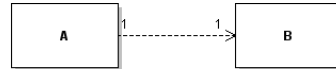
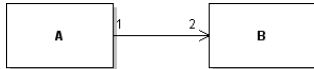


73

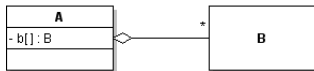
## NESNELER ARASINDAKİ İLİŞKİLER

### İLİŞKİLERDEKİ GİZLİ BİLGİLER

- Sahiplik, kullanma, parça-bütün ilişkileri oklarla gösterilmişse, sınıfların içerisinde ayrıntılı olarak gösterilmek zorunda değildir.



Şekildeki üç ilişki grubunda  
üstteki ilişkiler kapalı/gizli,  
alttaki ilişkiler açık olarak gösterilmiştir.

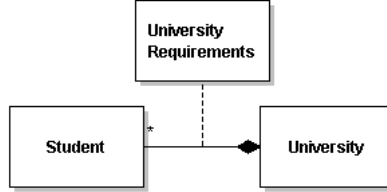


74

## NESNELER ARASINDAKİ İLİŞKİLER

### KURAL KOYAN (ASSOCIATIVE) SINIFLAR

- İki nesne arasındaki ilişkinin kuralları bir başka sınıf tarafından belirleniyorsa kullanılır.
- Gösterim:

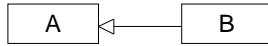


75

## NESNELER ARASINDAKİ İLİŞKİLER

### KALITIM

- Sınıf düzeyinde bir ilişkidir ve nesne düzeyinde etkileri vardır (anlatılacak).
- Kalıtım benzetmesi: Bir çocuk, ebeveyninden bazı genetik özellikleri alır.
- NYP: Mevcut bir sınıftan yeni bir sınıf türetmenin yoludur.
- Gösterim:



Kalıtım (inheritance)

- Ok yönüne dikkat!

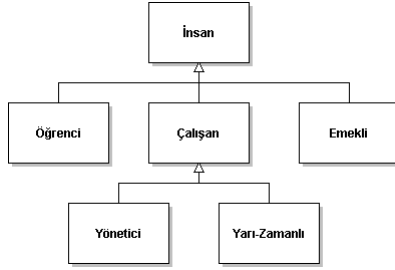
- A:
  - Ebeveyn sınıf (parent)
  - Üst sınıf (super)
  - Temel sınıf (base)
- B:
  - Çocuk sınıf (child)
  - Alt sınıf (sub)
  - Türetilmiş sınıf (derived)
- Kalıtımın işleyişi:
  - Kalıtım yolu ile üst sınıftan alt sınıfa hem üye alanlar hem de üye metotlar aktarılır (private üyeler hariç).

76

## NESNELER ARASINDAKİ İLİŞKİLER

### KALITIM

- Kalıtım kuralları:
  - Miras alma adlandırmasının uygunsuzluğu: Alt sınıf herhangi bir üyeyi miras almamayı seçemez.
  - Ancak kalıtımla geçen metotların gövdesi değiştirilebilir.
  - Alt sınıfta yeni üye alanlar ve üye metotlar tanımlanabilir.
  - Alt sınıflardan da yeni alt sınıflar türetilir. Oluşan ağaç yapısına kalıtım hiyerarşisi veya kalıtım ağacı denir.



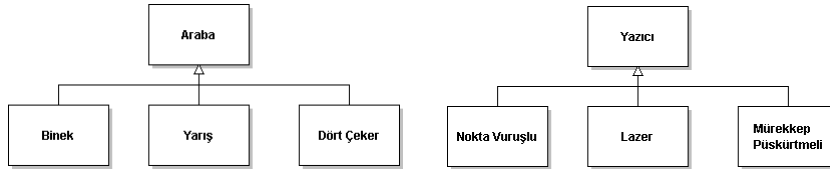
- Kalıtım ağacını çok derin tutmak doğru değildir (Kırılgan üst sınıf sorunu: Bina temelinin çürümesi gibi).

77

## NESNELER ARASINDAKİ İLİŞKİLER

### KALITIM

- Kalıtımın etkileri:
  - Genelleşme – özelleşme ilişkisi (generalization – specialization).
    - Alt sınıf, üst sınıfın daha özelleşmiş, daha yetenekli bir türüdür.
  - Yerine geçebilme ilişkisi (substitutability).
    - Alt sınıftan bir nesne, üst sınıftan bir nesnenin beklendiği herhangi bir bağlamda kullanılabilir.
    - Bu nedenle IS-A ilişkisi olarak da adlandırılır.

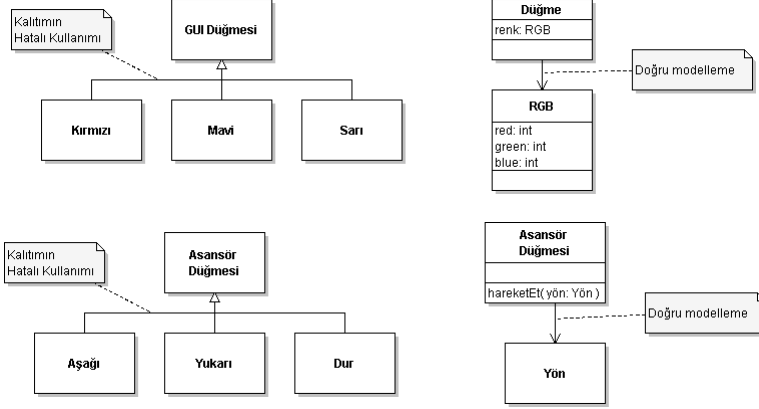


78

## NESNELER ARASINDAKİ İLİŞKİLER

### KALITIM

- Kalıtımın yanlış kullanımı:

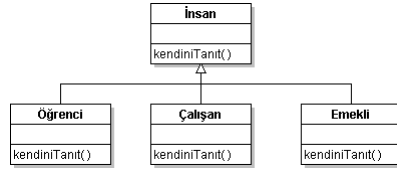


79

## NESNEYE YÖNELİK PROGRAMLAMADA ÖZEL KONULAR

### ÇOK BİÇİMLİLİK (POLYMORPHISM)

- Kalıtımla geçen metodların gövdesinin değiştirilebileceğini gördük.
- Üst sınıftan bir nesnenin beklendiği her yerde alt sınıftan bir nesnenin kullanılabileceğini gördük.



- kendiniTanıt() metodunun alt sınıflarda gösterilmesine gerek yoktu ama vurgulamak için yaptım.

- İnsan türünden bir dizi düşünelim, elemanları İnsan ve alt sınıflarından karışık nesneler olsun. Dizinin tüm elemanlarına kendini tanıt dediğimizde ne olacak?
  - Çalışma anında doğru sınıfın metodu seçilir.

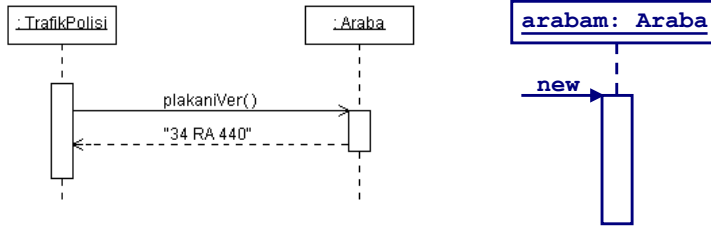
80



## NESNEYE YÖNELİK PROGRAMLAMANIN TEMELLERİ

### NESNELER VE SINIFLAR

- Nesneler, mesajlar ile çalışır.
  - Bir nesnenin bir eylemini çalıştırmak, o nesneye bir mesaj göndermektir.
  - Bir nesneye yönelik program, nesneler arasındaki mesaj akışları şeklinde yürür.
  - Kurucu metot (constructor): Nesneyi oluşturma mesajı.
  - Sonlandırıcı / Yok edici metot (destructor): Nesnenin yaşamını sona erdirmesi mesajı.



81

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### GEREKSİNİMLERİN BELİRLENMESİ

- Gereksinimler belgesi:
  - Müşterinin programdan beklentilerini anlatan, doğal konuşma dili ile yazılmış belge.
- Örnek gereksinimler belgesi:

#### NextGenPOS Perakende Satış Programı

Eski yazılım ihtiyaçlarımızı karşılayamadığından, yenilenecek donanım ile birlikte perakende satış programımızın da yenilenmesine gerek duyuyoruz. Program kasada yapılan alış-veriş işlemlerine yardımcı olmalıdır. Yapılan her işlem program tarafından saklanmalı; mali bilgiler harici bütçe sistemine, mal çıkış bilgileri ise harici envanter sistemine iletilmelidir. Saklanan işlemler üzerinde daha sonra raporlamalar ve analizler yapılabilir. Sistem yapılan alış-verişler karşılığında müşteriye fiş vermelidir. Yapılan her satış için KDV de hesaplanarak ayrıca belirtilmelidir. Şirketimizin birden fazla şubesi olup tüm şubelerdeki işlemler merkezi sunucuya iletilmelidir.

- Doğal dille yazılmış gereksinimler belgesinden kullanım öykülerine geçiş yapılır.

82

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### GEREKSİNİMLERİN BELİRLENMESİ

- Kullanım öyküleri:
  - Programın yapacağı işleri ayrıntılı adımlarla ve belli kurallara uyarak anlatan belgeler.
- Kullanım öykülerinin oluşturulmasındaki amaç:
  - Ürünün sağlaması beklenen işlevleri ve ürünün çalışma ortamını belirlemek,
  - Son kullanıcı ve yazılım ekibi arasında bir anlaşma zemini belirlemek,
  - Son kullanıcı ve sistemin birbirleri ile nasıl etkileşimde bulunacağını açık ve belirsizlikten uzak olarak tanımlamak,
  - Doğrulama testleri için bir zemin oluşturmak.
- Bir kullanım öyküsünün bölümleri:
  - Giriş bölümü: Sistemin neyi hangi koşullar ve sınırlar içerisinde yapması gerektiğini anlatır.
  - Ana senaryo / Ana başarılı akış: Her şeyin yolunda gitmesi halinde yürütülecek eylemler.
  - Alternatif senaryolar: Bir aksilik olması halinde yapılacak işlemler.

83

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### KULLANIM ÖYKÜSÜ: Satış İşlemi

**Birincil Aktör:** Kasiyer.

**İlgililer ve İlgili Alanları:**

- Kasiyer: Doğru ve hızlı giriş ister, kasa açığı maaşından kesildiğinden ödeme hataları istemez
- Satıcı: Satış komisyonlarının güncellenmesini ister
- Müşteri: En az çaba ile hızlı hizmet ister. Mal iadesinde kullanmak üzere fiş ister.
- ...

**Ön Koşullar:**

- Kasiyerin kimliği doğrulanır.

**Son Koşullar:**

- Ödeme tahsil edilir. Satış kaydedilir. Fiş yazılır.

- Dikkat: Kullanım öyküsünde yer alacak her şey, verilen ilgi alanlarına giren şeyler olmalıdır.
- Aktör: Sistem ile etkileşimde bulunan varlıklar.
  - İnsan
  - Yazılım veya donanım.

84

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### KULLANIM ÖYKÜSÜ: Satış İşlemi

#### Ana Öykü:

1. Müşteri kasaya alacağı mallarla gelir.
2. Kasiyer yeni bir satış işlemi başlatır.
3. Kasiyer ürünün barkodunu girer.
4. Sistem bir satış kanalı maddesi oluşturur. Bu maddede ürün tanımı, fiyatı ve toplam bedel (aynı maldan birden fazla alınmış olabilir) yer alır.
5. Kasiyer 3. ve 4. adımları müşterinin alacağı tüm mallar için tekrarlar.
6. Sistem toplam bedeli vergi iadesi ile birlikte hesaplar.
7. Kasiyer müşteriye toplamı bildirir ve ödeme ister.
8. Müşteri ödemeyi yapar ve sistem ödemeyi tahsil eder.
9. Sistem tamamlanan işlemin kaydını tutmayı tamamlar ve harici envanter ile mali sistemlere gerekli bilgileri gönderir.
10. Sistem makbuz verir.
11. Müşteri mallarla birlikte ayrılır.

85

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### KULLANIM ÖYKÜSÜ: Satış İşlemi

#### Alternatif Öyküler:

##### 3a. Geçersiz barkod

1. Sistem uyarı mesajı verir ve kayıt girişini reddeder.

##### 3-7a. Müşteri bir kalem mali alışverişten çıkartmak ister.

1. Kasiyer satıştan çıkarmak üzere ürünün barkodunu okutur.
2. Sistem güncel toplamı bildirir.

...

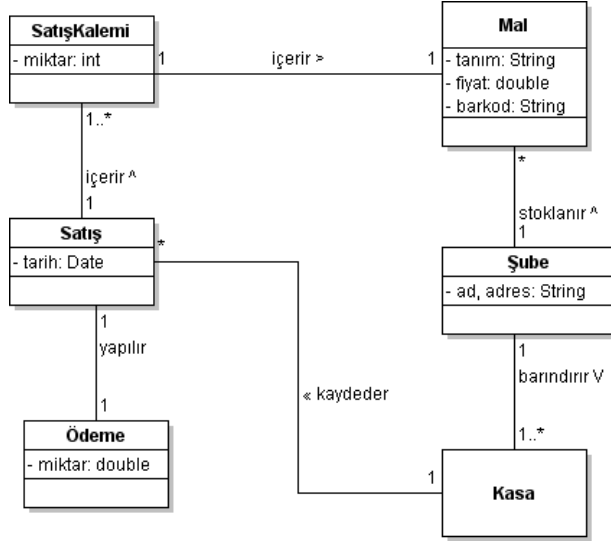
### KULLANIM ÖYKÜLERİNİN GRAFİK GÖSTERİMİ

- Kullanım öyküleri, ayrıntılı ve uzun belgelerdir.
- Yazılımın yapacağı işlerin özet gösterimi için kullanım şemaları çizilir (use-case diagrams).
- Çizim kurallarını verdikten sonra örnek öykünün şemasını çiz.

86

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### ÖRNEK ALAN MODELİ



87

## NESNEYE YÖNELİK ÇÖZÜMLEME SÜRECİ

### ÇÖZÜMLEME SÜRECİNİN BELGELENDİRİLMESİ

- Bir nesneye yönelik programın çözümleme sürecinin belgelendirilmesinde yer alan önemli belgeler:
  - UML Kullanım şemaları,
  - Kullanım senaryoları,
  - UML sınıf şemaları,
- Veritabanı işlemleri yapılacaksa bunlara ek olarak:
  - E-R diyagramı

88

**YAZILIM MÜHENDİSLİĞİ DERS NOTLARI**  
**Yrd.Doç.Dr. Yunus Emre SELÇUK**

**NESNEYE YÖNELİK TASARIM SÜRECİ**

89

**NESNEYE YÖNELİK TASARIM SÜRECİ**

**ANALİZDEN TASARIMA GEÇİŞ**

- Tasarım sırasında çizdiğimiz çeşitli şemalar ve hazırladığımız sözleşmeler, analiz sırasında oluşturduğumuz çeşitli şemalar ve metinleri ayrıntılandırır ve/veya değiştirir.
- Yazılım geliştirme sırasında kod dışında ortaya çıkardığımız her türlü metin ve şemaya "artefact" denilmektedir.

90

## NESNEYE YÖNELİK TASARIM SÜRECİ

### GİRİŞ

- Nasıl? sorusuna yanıt aranır.
- Nesne modeli: Analizden tasarıma.
  - Doğrudan problem alanı ile ilgili nesnelerden oluşan model, yardımcı nesnelerle zenginleştirilir.
- Ana işlem grupları:
  - Nesne tasarımı: Problem alanı ile ilgili nesneler
  - Sistem tasarımı: Alt yapıyı ve gereçleri oluşturan nesneler
- Sistem katmanında bulunabilecek bileşenler:
  - Yazılım mimarisi: İstemci - sunucu, eşler arası, olay tabanlı, vb.
  - Kullanıcı arayüzü
  - Veri yönetimi
  - Ağ programlama
- Sistem katmanını çoğunlukla kendimiz sıfırdan oluşturmayız, hazır altyapı programlarını (framework) kullanırız.

91

## NESNEYE YÖNELİK TASARIM SÜRECİ

### TASARIM ÖLÇÜTLERİ

- Tasarım ölçütleri:
  - Ayrılabilirlik: Anlamlı parçalara ayrılabilme.
    - Parça: Sınıf/sınıf grubu.
    - Üstünde çalıştığımız problem hangi düzeyde alt problemlere bölünebiliyorsa, tasarımı da aynı düzeyde ayrıştırılabilir.
  - Birleştirilebilirlik: Bir parçanın başka tasarımlarda da kullanılabilecek şekilde diğer parçalarla birleştirilebilmesi.
  - Anlaşılabilirlik: Bir parçanın diğer parçalar hakkında bilgiye gerek duyulmadan anlaşılabilmesi.
  - Süreklilik: Yapılacak küçük değişikliklerin etkilerinin en az sayıda parçaya yayılması (tercihen tek sınıfa).
  - Koruma: Olası hataların düzeltilmesine yönelik büyük değişikliklerin etkilerinin geniş bir alana yayılmasının önlenmesi.

92

## NESNEYE YÖNELİK TASARIM SÜRECİ

### TASARIM İLKELERİ

- İyi bir tasarıma götüren iki temel ilke:
  - Düşük bağlaşım (Low coupling)
  - Yüksek uyum (High cohesion)
- Bu ilkeler hem birbirlerine hem de uygulama alanına bağımlıdır.
- Başka ilkeler de öne sürülebilir, ancak bu ikisi en temel ölçütlerdir.

93

## NESNEYE YÖNELİK TASARIM SÜRECİ

### DÜŞÜK BAĞLAŞIM – LOW COUPLING

- Bağlaşım: Bir parçanın diğer parçalara bağımlılık oranı.
  - Parça: Sınıf, alt sistem, paket
- Bağımlılık: Bir sınıfın diğerinin:
  - Hizmetlerinden yararlanması,
  - İç yapısından haberdar olması,
  - Çalışma prensiplerinden haberdar olması,
  - Özelleşmiş veya genelleşmiş hali olması (kalıtım ilişkisi).
- Çeşitli sınıf şemaları ile bağlaşım soruları sor.
  - İlişkide bulunulan diğer sınıfların sayısı arttıkça bağlaşım oranı artar.
- Düşük bağlaşımın yararları:
  - Bir sınıfta yapılan değişikliğin geri kalan sınıfların daha azını etkilemesi,
  - Yeniden kullanılabilirliğin artması

94

## NESNEYE YÖNELİK TASARIM SÜRECİ

### YÜKSEK UYUM – HIGH COHESION

- Uyum: Bir parçanın sorumluluklarının birbirleri ile uyumlu olma oranı.
- Yüksek uyumun yararları:
  - Sınıfın anlaşılma kolaylığı artar.
  - Yeniden kullanılabilirlik artar.
  - Bakım kolaylığı artar
  - Sınıfın değişikliklerden etkilenme olasılığı düşer.
- Genellikle:
  - Düşük bağlaşım getiren bir tasarım yüksek uyumu,
  - Yüksek bağlaşım getiren bir tasarım ise düşük uyumu beraberinde getirir.

95

## NESNEYE YÖNELİK TASARIM SÜRECİ

### TASARIMIN BELGELENDİRİLMESİ

- Bir nesneye yönelik programın tasarım sürecinin belgelendirilmesinde yer alan önemli belgeler:
  - Ayrıntılı UML sınıf şemaları: Vazgeçilmez.
- Tasarımın ihtiyaçlarına göre alttaki diğer belgelerin çeşitli bileşimleri de kullanılabilir:
  - Sözleşmeler
  - UML Etkileşim şemaları (Interaction diagrams)
    - Sıralama şemaları (Sequence diagrams)
    - İşbirliği şemaları (Collaboration diagrams)
  - UML Etkinlik şemaları (Activity diagrams)
  - UML Durum diyagramları (State diagrams)

96



## NESNEYE YÖNELİK TASARIM SÜRECİ

### SÖZLEŞME İLE TASARIM – DESIGN BY CONTRACT

- Sözleşme:
  - Kullanım senaryosunun ayrıntılandırılması ile elde edilir.
  - Bir nesnenin bir eylemi, bir sözleşme ile ayrıntılandırılır.
  - Her metota sözleşme yazılacak diye bir koşul yoktur.
    - Zaten kolay anlaşılabilir metotların sözleşmeye ihtiyacı yoktur.

97

## NESNEYE YÖNELİK TASARIM SÜRECİ

### SÖZLEŞME İLE TASARIM – DESIGN BY CONTRACT

- Örnek sözleşme:

#### Sözleşme No: 2 – Satış Kalemi Girişi

<b>İşlem:</b>	ürünGir( barkod: Barkod, adet: integer )
<b>Çapraz Başvurular:</b>	Satış kullanım senaryosu
<b>Ön Koşullar:</b>	Süregelen bir satış işleminin olması
<b>Son Koşullar:</b>	<ul style="list-style-type: none"><li>- Bir SatışKalemi örneği olan satisKalemi oluşturulmuştur.</li><li>- satisKalemi süregelen satis ile (Satış örneği) ilişkilendirilmiştir.</li><li>- satisKalemi.adet üyesine o malın satış adedi atanmıştır.</li><li>- satisKalemi, satılan mal ile uyuşan barkod sayesinde bir UrunTanımı örneği ile ilişkilendirilmiştir.</li></ul>

98

## NESNEYE YÖNELİK TASARIM SÜRECİ

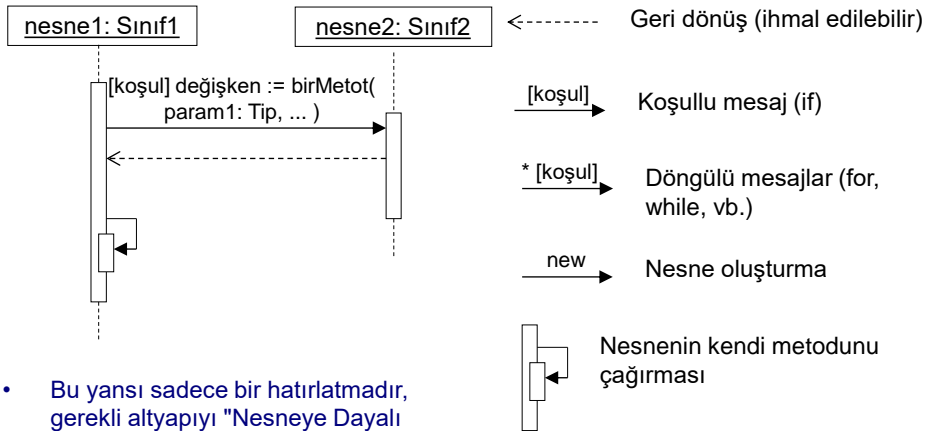
### SÖZLEŞME İLE TASARIM – DESIGN BY CONTRACT

- Sözleşmeler hakkında bazı ayrıntılar:
  - Ön koşullar tüm sistem hakkındaki bilgilerdir.
  - Son koşullar sadece problem alanı ile ilgili nesnelerin durum değişiklikleri hakkındadır.
  - Sözleşmeler her zaman gerekli olmayabilir.
  - Son koşullarda edilgen geçmiş zaman kullanılması, bunların işlemin sonunda tamamlanmış eylemler olduğunu vurgulamak açısından yerinde olacaktır.
  - Sözleşme içerisinde ilişkilerin kurulmasını belirtmeyi unutmayın!
  - Sözleşme yazılması problem alanı çözümlemesinde güncellemelere yol açabilir.

99

## SIRALAMA ŞEMALARI AYRINTILARI

### SIRALAMA ŞEMALARI



- Bu yansı sadece bir hatırlatmadır, gerekli altyapıyı "Nesneye Dayalı Kavramlar" dersinde kazanmıştınız.

100

## ETKİNLİK ŞEMALARI – ACTIVITY DIAGRAMS

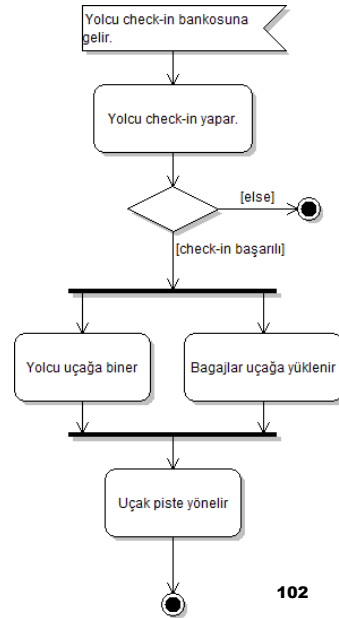
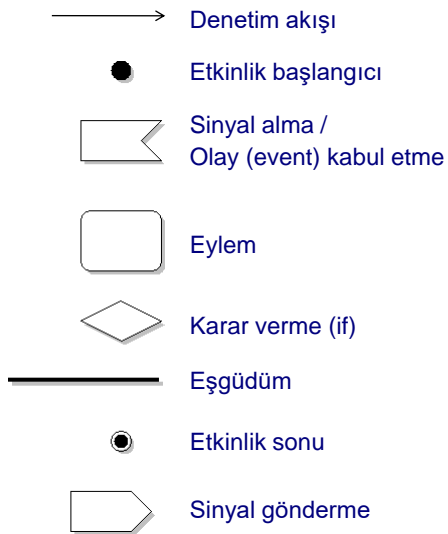
### UYGULAMA ALANLARI

- Denetim akışını olaylar üzerinden göstermeye yarar.
- İş kurallarını göstermek ve paralel çalışma (multithreaded) ayrıntılarına girmek gerektiğinde, etkileşim şemalarından daha yararlıdır.

101

## ETKİNLİK ŞEMALARI – ACTIVITY DIAGRAMS

### ÇİZİM KURALLARI ve ÖRNEK ÇİZİM

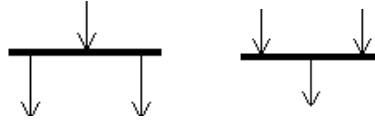


102

## ETKİNLİK ŞEMALARI – ACTIVITY DIAGRAMS

### ÇİZİM BİLGİLERİ

- Etkinlik şemaları başlangıç işareti veya sinyal alma işareti ile başlar.
- Sinyal alma: Beklemelidir.
  - Akış, bir sinyal alana kadar bekler.
  - Zamanlı olaylar da (timer) bununla gösterilebilir.
- Eşgüdüm: Beklemelidir.
  - Eşgüdüm çizgisine varan akış, çizgiyi geçmeden önce diğer akışların hepsini bekler.
  - fork
  - join

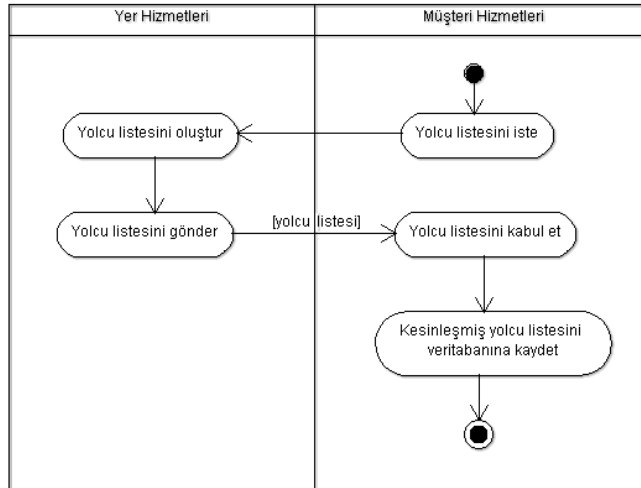


103

## ETKİNLİK ŞEMALARI – ACTIVITY DIAGRAMS

### ÖRNEK ÇİZİM

- Birden fazla aktörün ve aktörler arası bilgi akışının gösterilmesi:



104

## DURUM ŞEMALARI – STATE DIAGRAMS

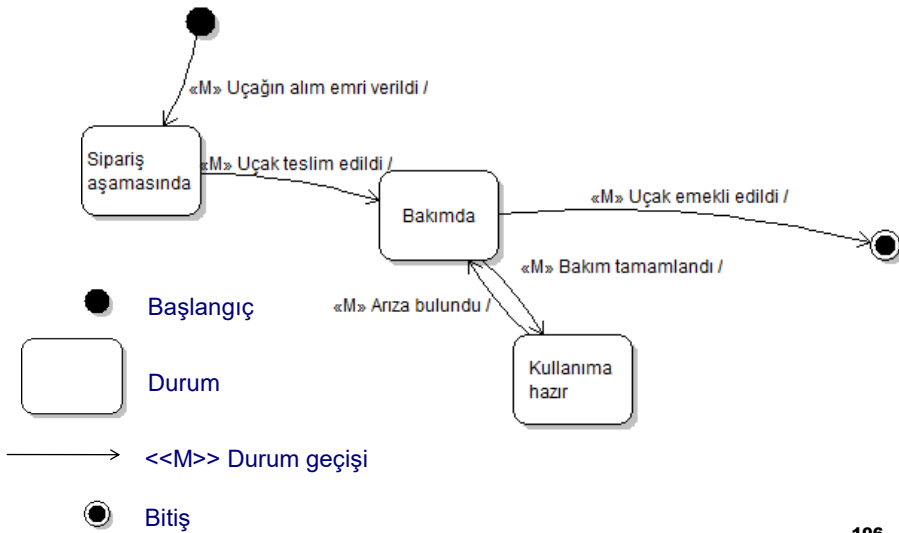
### UYGULAMA ALANLARI

- Bir varlığın içinde bulunabileceği durumları ve bu durumların birinden diğerine geçiş yapma kurallarını anlatmaya yarar.

105

## DURUM ŞEMALARI – STATE DIAGRAMS

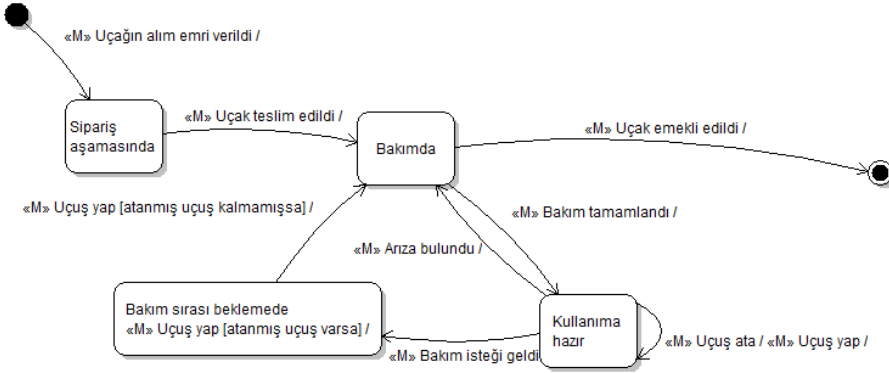
### ÇİZİM KURALLARI ve ÖRNEK ÇİZİM



106

## DURUM ŞEMALARI – STATE DIAGRAMS

### ÖRNEK ÇİZİM (2)



- Yorumlama: Kullanıma hazır bir uçak için bakım isteği gelmişse, uçak önce bakım sırasına alınır. Bu sırada önceden planlanmış uçuşları varsa onları yapar. Planlanan uçuşlar bitince uçak bakıma alınır.

107

## DURUM ŞEMALARI – STATE DIAGRAMS

### ÖRNEK ÇİZİM (2)

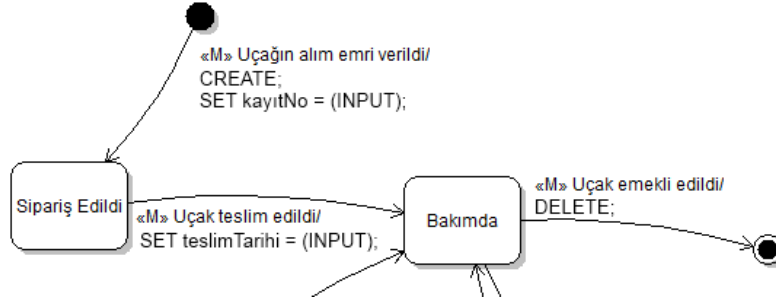
- Dezavantaj: Tutarlılık denetimi zor olabilir.
  - Karmaşık şemalarda mesajları takip etmek zorlaşır
  - Çünkü aynı mesaj birden fazla durumla ilişkili olabilir
    - Bu durumda aynı mesaj birden fazla yerde geçer.
  - Ör: Uçuş Yap mesajı.

108

## DURUM ŞEMALARI – STATE DIAGRAMS

### ÖRNEK ÇİZİM (3)

- Durum geçişi sırasında işlenen komutların şemada gösterilmesi:



109

## TASARIM MODELİ

### NextGenPOS Tasarım Modeli

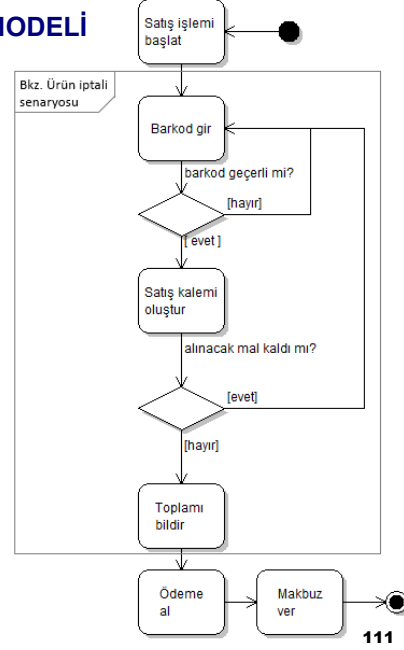
- Kullanım senaryosu metni ve alan modelinden yola çıkarak tasarım modelini oluşturalım.
  - Bu amaçla bir etkinlik, bir durum ve bir sınıf şeması çizelim.
  - Belki bu sırada keşfedeceğimiz yeni ayrıntılar olacaktır.
  - Tasarım modelindeki sınıf şemasının farkı, artık yönsüz ilişki bırakılmaması ve sınıfların metotlarının da eklenmesidir.

110

## TASARIM MODELİ

### NextGenPOS Tasarım Modeli

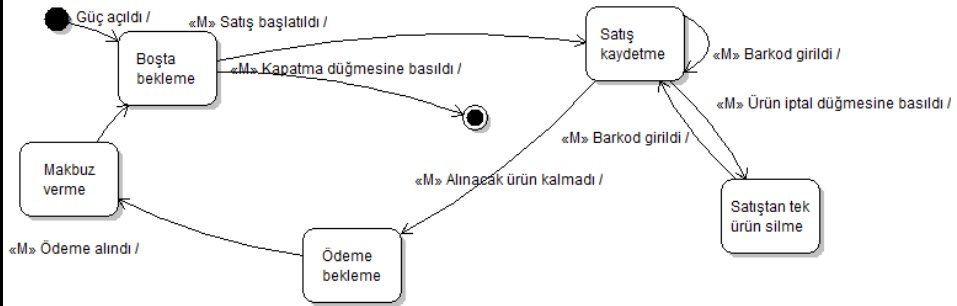
- Etkinlik şeması:



## TASARIM MODELİ

### NextGenPOS Tasarım Modeli

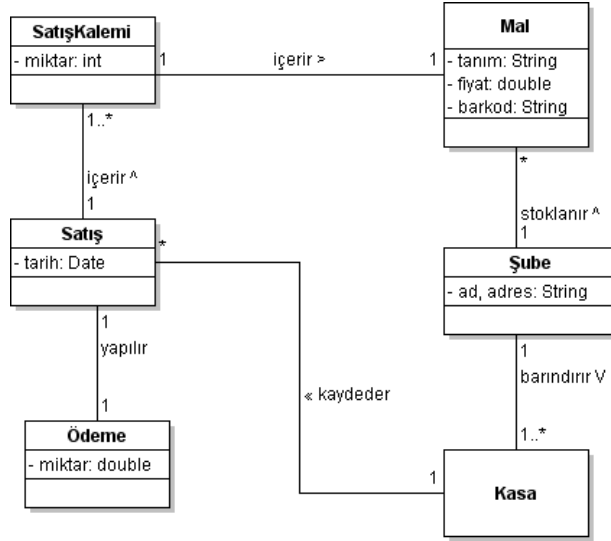
- Kasa ile ilgili durum şeması:





## NESNEYE YÖNELİK TASARIM SÜRECİ

- NextGenPos projesi alan modelini hatırlayarak tasarım modelimizi oluşturmaya çalışalım:

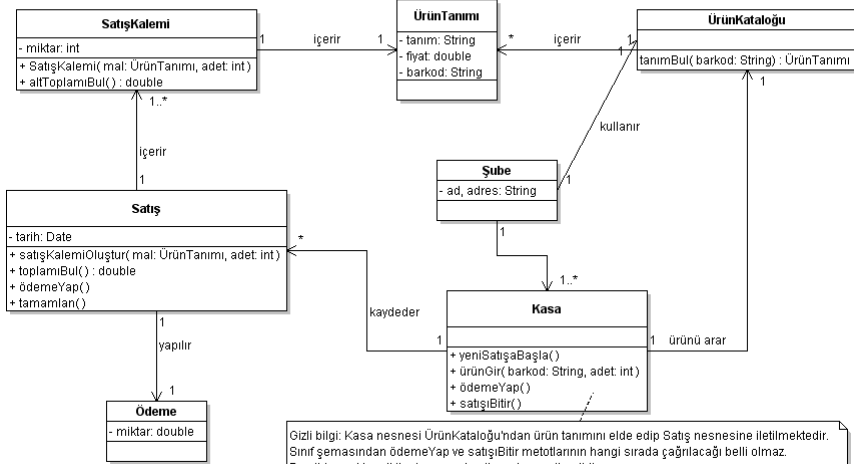


113

## NESNEYE YÖNELİK TASARIM SÜRECİ

### TASARIM MODELİ

Çözümleme aşamasında tespit edilen Mal sınıfının, sürecin ilerleyen aşamalarında, ikiye bölünmesine karar verilmiştir. Alternatif senaryolara yer verilmemiştir.



Gizli bilgi: Kasa nesnesi ÜrünKatalogu'ndan ürün tanımını elde edip Satış nesnesine iletilmektedir. Sınıf şemasından ödemeYap ve satışBitir metotlarının hangi sırada çağrılacağı belli olmaz. Bu gibi ayrıntılar etkileşim şemaları ile açıkça verilmelidir.

14

## YAZILIM MÜHENDİSLİĞİ TEMELLERİ DERS NOTLARI

Yrd.Doç.Dr. Yunus Emre SELÇUK

### YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

115

### YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

#### GENEL BİLGİLER

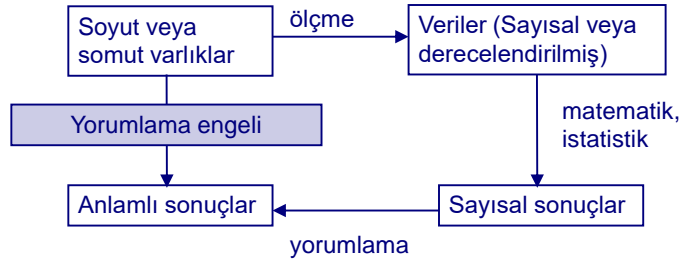
- Ölçme (Measuring): Somut veya soyut bir varlığın sahip olduğu bir özelliğini, sayısal veya derecelendirilmiş bir veri olarak ifade etmek.
  - Benim boyum 163 santimetredir.
  - Hava bugün 22 santigrat derecedir.
  - İlk ara sınav çok zordu.
- Ölçüt (Metric): Varlığın ölçülecek özelliğini ölçme biçimi.
  - Mesafe ölçütleri: Bir labirentteki Öklid ölçütü (Pisagor teoreminden) ve kuş uçuşu ölçütü.
  - Sıcaklık ölçütü: Santigrat ve Fahrenheit
- Ölçüm (Measurement): Belli bir ölçüte göre yapılan ölçme eyleminin sonucu.
- Ölçme/ölçüt/ölçüm karışıklığı
  - İngilizce'de daha da zor
    - Measure –ment ve –ing son eklerini ben özellikle koydum
  - Türkçe'de daha kolay
    - Yine de neyin isim, neyin sıfat, neyin eylem olduğunu karıştırmamalı.
- Neden ölçeriz?
  - Gerçek dünya ile ilgili, işimize yarayacak, anlamlı sonuçlar elde etmek için.

116

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### GENEL BİLGİLER

- Yorumlama engeli (Intelligence barrier):
  - Ölçmenin sonucu, aradığımız sonuçları elde etmek için doğrudan bir yol sunmayabilir,
  - ya da yapacağımız yorumlama zor olabilir.
  - Örnek: Otostopçunun galaksi rehberi'nde hayatın anlamı: 42!



117

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### YAZILIM ÖLÇÜMÜ

- Yazılım ölçümü zordur:
  - Bir başka deyişle, yorumlama engeli yüksektir.
  - Zorluğun nedenleri:
    - Yazılımın karmaşıklığı
    - Ölçütlerin nicel doğası
- Yazılımı neden ölçeriz?
  - Ne kadar iyi bir ürün ortaya çıkardığımızı anlamak
  - Ne kadar iş yapacağımızı kestirmek
  - Böylece ne kadar zaman ve para harcayacağımızı anlamak
  - Ölçülemeyen ilerleme yönetilemez: Proje yönetiminde yazılım ölçütleri kullanılır.

118

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### YAZILIM KALİTE ÖLÇÜTLERİ

- Nicel kalite ölçütleri farklı kişilerce farklı şekillerde öbeklenmekte ve farklı dallara ayrılmaktadır.
- ISO 9126 kalite ölçütleri:
  - İşlevsellik
    - Uygunluk, doğruluk, güvenlik, ...
  - Güvenilirlik
    - Olgunluk, hata bağıışıklığı, ...
  - Kullanılabilirlik
    - ...
  - Verimlilik/Etkinlik
    - ...
  - Bakım kolaylığı
    - ...
  - Taşınabilirlik
    - ...
- McCall ve arkadaşlarının kalite ölçütleri:
  - İşlevsel ölçütler
    - Doğruluk, Güvenilirlik, Bütünlük, Kullanılabilirlik, Verimlilik
  - Değişirtilme ölçütleri
    - Bakım kolaylığı, Esneklik, Sınanabilirlik
  - Taşınma ölçütleri
    - Taşınabilirlik, Yeniden Kullanılabilirlik, Birlikte Çalışabilirlik
- McConnell'a göre kalite ölçütleri:
  - İç kalite ölçütleri
  - Dış kalite ölçütleri

119

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### YAZILIM KALİTE ÖLÇÜTLERİ

- Dış kalite ölçütleri: Yazılımı kullananları ilgilendiren ölçütler.
  - Doğruluk(Correctness): Yazılımın hatalar içermemesi, gereksinimlerde belirtildiğı şekilde çalışması.
  - Etkinlik(Efficiency): Bellek ve işlemci gibi sistem kaynaklarının en az oranda kullanımı.
  - Güvenilirlik(Reliability): Sistemin her koşulda istenildiğı gibi çalışması, hatalar arasındaki ortalama zaman aralığının (MTBF) yüksek olması.
  - Güvenlik(Security): İzinsiz ve yetkisiz işlemler mümkün olmamalı.
  - Bütünlük(Integrity): Veriler ve işlemler arasındaki tutarlılığın korunması.
  - Uyarlanabilirlik(Adaptability): Sistemin değışik uygulamalar veya ortamlarda kullanılabilmesi için mümkün olduğunca az değışiklik gerektirmesi.
  - Hassaslık (Accuracy): Sistemin kendisinden beklenen işi mümkün olduğunca iyi yapabilmesi.
  - Sağlamlık(Robustness): Aykırı girişlere veya güç çalışma ortamlarına karşılık sistemin çalışmayı sürdürebilmesi.
  - Kullanılabilirlik(Usability): Yazılım kolay kullanılabilir olmalıdır.
  - ...
- Bu ölçütler örtüşebilir, bazı durumlarda birbirinden daha iyi veya daha zor ayrılabilir.

120

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### YAZILIM KALİTE ÖLÇÜTLERİ

- İç kalite ölçütleri: Yazılımı geliştirenleri ilgilendiren ölçütler.
  - Yeniden kullanılabilirlik(Reusability): Sistemin parçalarının başka sistemlerde kullanılabilmesinin kolaylığı.
  - Bakım kolaylığı (Maintainability): Yazılıma yeni yetenekler eklemenin, yazılımdaki hataları gidermenin veya yazılımın başarımını attırmanın mümkün olduğunca kolay olması.
  - Esneklik(Flexibility): Yazılımın orijinal olarak tasarlandığı uygulamanın dışında çalışabilmesi için gerekli olan değişikliklerin olduğunca az olması.
  - Taşınabilirlik(Portability): Yazılımın farklı donanım ve işletim sistemleri gibi değişik çalışma ortamlarına kolaylıkla aktarılabilmesi.
  - Okunabilirlik(Readability): Kodun kaynak kodunun incelenmesinin kolay olması.
  - Anlaşılabilirlik(Understandability): Yazılımın sistem, bileşen ve kod düzeylerinde anlaşılabilirliğinin mümkün olduğunca kolay olması. Okunabilirlik sadece kod düzeyinde anlaşılabilirliği sağlar.
  - Sınanabilirlik(Testability): Sistemin istenen gereksinimleri karşılayıp karşılamadığının sınanabilmesinin bileşen ve tüm sistem çapında mümkün olduğunca kolay olması.

121

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### ÖLÇME İLKELERİ

- Ölçme eyleminin içermesi gereken adımlar:
  - Tanımlama (Formulation): Ölçütler ölçülecek yazılıma uygun bir şekilde tanımlanır
    - Kullanılan yaklaşım: Yapısal programlama, NYP, vb.
    - Yazılımın türü: Gerçek zamanlı, gömülü, uygulama, vb.
  - Toplama (Collection): Tarif edilen ölçütlerin gerektirdiği verileri elde etme.
  - Hesaplama (Analysis): Ölçütlerin hesaplanması = Ölçümlerin elde edilmesi.
    - Matematiksel araçlar kullanılabilir.
    - Hesaplama mümkün olduğunca otomatik yapılmalıdır.
  - Yorumlama (Interpretation): Elde edilen ölçüm değerlerinden yararlı anlamlar çıkartılması.
  - Geri besleme/Kullanma (Feedback): Çıkarılan sonuçların yazılım ekibine bildirilmesi ve ekibin sonuçları kullanarak yazılımı iyileştirmesi.

122

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### ÖLÇME İLKELERİ

- Bir ölçütün sahip olması arzu edilen özellikler:
  - Uygun matematiksel özelliklere sahip olmalı:
    - Anlamlı bir ölçekte olmalı. Ör. 0-1 arası sonuçlar üretmeli.
    - Doğru (veya ters) orantıya sahip olmalı. Sonucun yükselmesi, ölçülen özelliğin iyi bir sonuca doğru ilerlemesi (gerilemesi) anlamına gelmeli.
  - Deneysel olarak doğrulanabilmeli
    - Doğrulanmasının ardından kullanılmalı.

123

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### ÖNERİLEN YAZILIM ÖLÇÜTLERİ

- Nesneye yönelik ölçütler:
  - Kaliteli bir yazılıma götüren tasarım ilkelerine yöneliktirler.
  - NYP'de çözümleme ve tasarım arasında kopukluk olmadığı için, aynı ölçütler çözümleme ve kodlama aşamalarında da kullanılabilir.
  - Böylece yazılım ekibi, 'kaliteli bir ürüne giden yolda' iz üstünde olup olmadıklarını anlayabilir.
  - Proje yöneticisi de, başka ölçütlerle birlikte, kestirimlerde bulunabilir.
- Chidamber ve Kemerer'in ölçütleri (CK metrics suite):
  - WMC: Sınıftaki ağırlıklı metod sayısı (Weighted Methods per Class).
  - DIT: Kalıtım ağacının derinliği (Depth of Inheritance Tree).
  - NOC: Alt sınıf sayısı (Number of Children)
  - RFC: Sınıfın yanıt kümesinin eleman sayısı (Response For a Class)
  - CBO: Sınıflar arası bağlaşım (Coupling Between Objects)
  - LCOM: Uyum eksikliği (Lack of COhesion in Methods)

124

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### CK ÖLÇÜTLERİ ÖRNEĞİ:

- WMC:
  - C1 sınıfının M1...Mn metotlarının karmaşıklıkları c1..cn.
$$WMC = \sum_{i=1}^n c_i$$
- Eleştiriler:
  - Metot karmaşıklığı neye göre belirlenecek?
  - Belirlemedeki öznellik güçlü yön mü, zayıf yön mü?
- Ölçütün rehberliği:
  - Bir sınıfın karmaşıklığını belirler.
  - Çok sayıda metodu olan sınıf:
    - Çok fazla sorumluluk yüklenmiştir, dağıtılması uygun olabilir.
    - Yüksek uyumun olup olmadığına tekrar bakılmalıdır.
    - Uygulamaya özeldir, yeniden kullanılabilirliği düşüktür.

125

## YAZILIM KALİTESİ VE YAZILIM ÖLÇÜTLERİ

### DİĞER KALİTE ÖZELLİKLERİNE YÖNELİK ÖLÇÜTLER

- Bazı bağlaşım ölçütleri
  - COMIAS
  - CBMC
- Sınanabilirlik ölçütleri
  - Halstead ölçütleri (Tartışmalı)
  - Binder'in seçtiği ölçütler
- Bakım kolaylığı ölçütleri
  - IEEE Std. 982.1-1998'de yazılım olgunluk ölçütü

### NE YAPILABİLİR?

- Ölçütlerin büyük çoğunluğu mükemmel değildir.
- Yine de bu zayıf noktalar genellikle çok özel durumlarda ortaya çıkar.
- Bu nedenle ölçütler kullanılmalı, ancak tabulaştırılmamalı, sadece (çok da hassas olmayan) bir rehber olarak kullanılmalı.
- Belli bir kalite ölçütüne yönelik olarak, şimdiye dek önerilen ölçütlerden bazıları seçilip, sezgisel olarak bir araya getirildikten sonra piyasada sınanarak iyileştirilebilir.
  - Sezgisel yetenek nasıl bulunacak?
  - Özel sektör gerekli çabaya nasıl ikna edilecek?

126

**YAZILIM MÜHENDİSLİĞİ DERS NOTLARI**  
**Yrd.Doç.Dr. Yunus Emre SELÇUK**

**YAZILIM PROJE YÖNETİMİNE GİRİŞ**

127

**YAZILIM PROJE YÖNETİMİNE GİRİŞ**

**GENEL BİLGİLER**

- Yazılım projeleri önemli oranda başarısızlığa uğramaktadır:
  - Yazılım geliştirmedeki zorluklar.
  - Ölçek büyüklüğünden kaynaklanan zorluklar: Yazılım ölçeği, kişi ölçeği, vb.
  - Kestirimdeki zorluklar.
  - İnsanlarla çalışmadaki zorluklar.
  - Teknolojideki değişimler.
  - Gereksinimlerdeki değişimler.
  - Politik değişimler.
  - Mali değişimler.
- Yazılım proje yönetimi, sayılan zorlukların çözümüne odaklanır.
- Önem sırasına göre proje yönetiminin ilgi alanları:
  - Kişiler
  - Ürün
  - Süreç
  - Proje

128



## YAZILIM PROJE YÖNETİMİNE GİRİŞ

### GENEL BİLGİLER

- Temel amacın kullanıcılara bir yarar sağlamak olduğunu hiçbir aşamada unutmayın.
- Planlama yaklaşımları:
  - Basitlik yaklaşımı: Geleceğin getireceği değişiklikler çoğu zaman ayrıntılı planlama gereğini ortadan kaldırır.
  - Geleneksel yaklaşım: Planlama proje için bir yol haritası belirler; harita ne kadar ayrıntılı ise kaybolma olasılığı o kadar düşer.
  - Çevik yaklaşım: Ön hazırlık gereklidir ancak asıl harita proje ilerledikçe çizilir.

129

## YAZILIM PROJE YÖNETİMİNE GİRİŞ

### PLANLAMA

- Planlama ilkeleri:
  - Projenin sınırlarını belirleyin: Nereye gideceğinizi bilmezseniz kaybolursunuz.
  - Müşteriyi planlama eylemlerine katın: Öncelikleri, sınırları ve zamanlamayı müşteri belirler (bu sırada) ancak gerçekçiliği korumak amacıyla yazılım ekibi pazarlık yapar (aksi sırada).
  - Planlamanın doğası yinelemelidir: Plan asla taşa yazılmaz.
  - Bildiklerinizi kullanarak kestirimlerde bulunun: Bilginin kapsamı, doğruluğu ve belirginliği kestirimin doğruluğunu etkiler.
  - Planın her aşamasına risk değerlendirmesini ekleyin: Teknik, mali, kişisel, politik riskler.
  - Gerçekçi olun: Yazılımcı süpermen veya robot değildir.

130

## YAZILIM PROJE YÖNETİMİNE GİRİŞ

### PLANLAMA

- Planlama ilkeleri (devam):
  - Planların ölçeği: İnce ayrıntılı planlar daha kısa vadeli, genel ayrıntılı planlar daha uzun vadeli. Zaman ilerledikçe genel ayrıntıdan ince ayrıntıya geçilir.
  - Kalite güvence eylemlerini tanımlayarak plana ekleyin.
  - Değişikliğin nasıl kabul edileceğini müşteri ile sözleşmeye bağlayın.
  - Planın gidişinden gözünüzü ayırmayın ve gerekli ayarlamaları yapın.

131

## YAZILIM PROJE YÖNETİMİ

### PROJE YÖNETİMİNDE KİŞİ ETKENLERİ

- Takım yöneticisi:
  - Teknik ekibin bir parçası olduğundan teknik yetenekleri yüksek olmalıdır.
  - Ağırlıklı olarak insanlarla ilgili eylemlerde bulunacağından, sosyal ve yönetsel yetenekleri de yüksek olmalıdır.
- İyi bir teknik yöneticinin özellikleri:
  - Teknik ekibi istekli kılabilir.
  - Kişileri ve yazılım geliştirme sürecini, üzerinde çalışılan ürüne/ürün parçasına göre düzenleyebilir.
  - Düzenleme küçük veya büyük ölçekte olabilir.
  - Ürün ve süreç sınırları belli olsa da, ekibini yaratıcı fikirler üretmeye teşvik edebilir.
  - İyi bir sorun çözücü olmalıdır.
    - Hem teknik hem de yönetsel sorunlarla uğraşabilir.
    - Sorunlara tanı koyabilmeli ve ortaya uygun bir çözüm koyabilir.
    - Seçilen çözüm tıkandığında ısrarcı olmamalıdır.
  - Sorumluluk alabilir.

132

## YAZILIM PROJE YÖNETİMİ

### PROJE YÖNETİMİNDE KİŞİ ETKENLERİ

- Yazılım geliştirme ekibi (teknik ekip):
  - Takım ruhuna uygun kişilerden oluşmalıdır:
    - Takım üyeleri birbirine saygı duymalıdır.
    - Takım üyeleri ortak amaç etrafında kenetlenmelidir.
    - Takım üyeleri birbirlerini tamamlayan yeteneklere sahip olmalıdır.
  - Takım ruhunu bozan etkenler:
    - Telaşlı iş ortamı.
    - Sık ortaya çıkan hayal kırıklıkları ve başarısızlıkların takım üyeleri arasındaki sürtüşmeyi artırması.
    - Doğru yönetilemeyen yazılım geliştirme süreci.
    - Takım yapısının ve rollerinin belirsiz tanımlanması.

133

## YAZILIM PROJE YÖNETİMİ

### PROJE YÖNETİMİNDE KİŞİ ETKENLERİ

- Takım yapıları:
  - Kapalı yaklaşım:
    - Geleneksel bir yetki hiyerarşisi ve kontrol mekanizmaları bulunur.
    - Geçmiş deneyimlere benzer projelerde başarılı bir yapıdır.
    - Yaratıcı fikirler ortaya çıkarmak için çok uygun değildir.
  - Rastgele (Random) yaklaşım: Serbest yaklaşım.
    - Takım üyelerinin bireysel ve teknik yeteneklerine göre kendi aralarında bir yapı kurmasıdır.
    - Yaratıcı fikirler ortaya çıkarmak için en uygun yaklaşımdır.
    - Disiplin elde etmek zor olabilir.
  - Açık yaklaşım: Kapalı ve rastgele arasında.
    - Kontrol mekanizmaları bulunur ancak yapılanma serbesttir.
    - Demokratik yapı.
    - Karmaşık sorunların çözümü için uygun.
    - Etkinliği (efficiency) sağlamak zor olabilir.

134

## YAZILIM PROJE YÖNETİMİ

### PROJE YÖNETİMİNDE KİŞİ ETKENLERİ

- Takım yapıları (devam)
  - Eşzamanlı (synchronous) yaklaşım:
    - Problemin takımın üzerine düşen bölümünün de alt parçalara ayrılabilirdiği durumlarda kullanılabilir.
    - Takım kendi içerisinde problemin alt parçalarını paylaşır.
    - Alt takımlar arasında etkileşim azdır.
- Takım içi ve takımlar arası haberleşme:
  - Resmi yollar: Yazı ile, zamanlı mesajlaşma ile, kurallı ve zamanlanmış toplantılar ile.
  - Gayrı resmi yollar: Sözlü iletişim, kişisel etkileşimler, gün içerisinde gerektiğinde.

135

## YAZILIM PROJE YÖNETİMİ

### PROJE YÖNETİMİNDE ÖLÇÜM

- Proje Ölçümü:
  - Yazılımın ölçülmesidir (İncelendi).
  - Odak: Teknik düzey.
  - Amaç: İç kalite ölçütlerini yüksek tutmak
  - Yöntem: Ölçüm sonuçlarına göre, yazılım geliştirme ekibini iyiye doğru yönlendirmek.
- Süreç Ölçümü:
  - Yazılım geliştirme sürecinin ölçülmesidir.
  - Odak: Yönetimsel düzey.
  - Amaç: Dış kalite ölçütlerine yöneliktir.
  - Yöntem: Sürecin tüm aşamalarında tutulan istatistiklere göre, hem teknik hem de yönetimsel açıdan süreçleri iyileştirmek.

136

## YAZILIM PROJE YÖNETİMİ

### PROJE YÖNETİMİNDE ÖLÇÜM

- Kalite ile ilgili etkenler:
  - Kişiler: Çalışanların yetenek ve istekliliği en önemli etkidir.
  - Ürün: Ürünün karmaşıklığı kaliteye olumsuz etki eder. Karmaşıklıkla artan zorluk düzeyi, takımın maneviyatını kırabilir.
  - Teknoloji: Yazılım geliştirme araçları ve donanım bileşenlerinin kalitesi.
    - Kalite: Olgunluk, yeterlilik, etkinlik, eğitimsel belgeler, vb.
  - Çevresel koşullar: Ana etkenlerle de ilişkilendirilebilir.
    - Müşterinin özellikleri: İletişim yeteneği, işbirliği yeteneği, vb.
    - İş koşulları: İş akışı kuralları, şirket kültürü, vb.
    - vb.
- Ölçüt toplama ve gizlilik düzeyleri:
  - Bireysel düzey: Kişiye özel olmalıdır.
  - Takım düzeyi: Takıma özel olmalıdır.
  - Proje düzeyi: Tüm çalışanlara özel olmalıdır.
- Gizlilik düzeyi tartışması:
  - Üstün, astlarının ölçütlerini izleyebilmesi ancak denklemin birbirlerinin ölçütlerini görememesi ise bir başka yaklaşımdır.
  - Amaç insanları utandırmak, övmek veya tehdit etmek olmamalı.

137

## YAZILIM PROJE YÖNETİMİ

### PROJE YÖNETİMİNDE ÖLÇÜM

- Süreç Ölçütleri:
  - Hata sayısı: Bulunan hata sayısı.
    - Bakım aşamasından önce ve sonra bulunan hataların sayısı, ayrı öneme sahiptir.
  - Hata giderme etkinliği:
    - $HGE = \frac{TÖ}{(TÖ + TS)}$
    - TÖ: Ürünün müşteriye tesliminden önce bulunan hata sayısı
    - TS: Teslimden sonra (bakım aşamasında) bulunan hata sayısı
    - TÖ ve TS yerine, yazılım geliştirme sürecinin ardışıl adımları olan A(i) ve A(i+1) kullanılabilir. Örneğin:
      - A(i) : Çözümlemede bulunan hata sayısı
      - A(i+1) : Tasarımda bulunan çözümleme kaynaklı hata sayısı.
  - Doğruluk: Bulunan hata sayısının proje boyutuna oranı.
  - Bakım kolaylığı: Bir hatanın bulunması ile giderilmesi arasında geçen zaman.
  - Güvenlik: Bulunan güvenlik açıkları, açıkların ciddilik düzeyi, vb. gibi veriler üzerinde yapılan ölçümler.
  - Ve diğer dış kalite ölçütleri...

138

## YAZILIM PROJE YÖNETİMİ

### PROJE YÖNETİMİNDE ÖLÇÜM

- Süreç ölçütlerinin kullanımı:
  - Ölçütler amaç değil, araç olmalı.
  - Şirketin boyutu ile ölçüm yapmaya ayrılan çaba orantılı olmalı.
  - Öncelikleri ve hedefleri belirleyip ona uygun ölçütler kullanılmalı.
    - Amaca yönelik ölçümler yapılmalı.
  - SEI (Software Engineering Institute) ve SPC (Software Productivity Center)'nin önerdiği ölçüt geliştirme yaklaşımları mevcuttur.
    - Ve daha başkaları...
  - Süreç iyileştirme modelleri çeşitli ölçütlerin kullanılmasını da gerektirir.

139

## YAZILIM PROJE YÖNETİMİ

### YAZILIM PROJELERİNİN VAZGEÇİLMEZ ARAÇLARI

- IDE'ler.
- UML modelleme araçları.
  - İki yönlü dönüşüm yeteneğine sahip olması (model ve kod arasında) tercih edilir.
- Sürümlendirme yazılımı (version control systems)
- Sınama yazılımı (testing framework)
  - Bir yapılandırma yazılımı (build system) ile tümleşik olması tercih edilir.
- İş kalemleri izleme yazılımı (work item tracking)
- Tüm araçların IDE tümleşik olması tercih edilir.
  - Java: IBM Rational Application Developer
  - .NET: Microsoft Team System

140

## YAZILIM PROJE YÖNETİMİ

### RİSK YÖNETİMİ

- Risk ile uğraşma taktikleri:
  - Sonradan (Reactive): Risk gerçekleşince çaresine bakmak.
  - Önceden (Proactive): Riskleri daha gerçekleşmeden önlemeye çalışmak.
  - Nasrettin hoca: Kızını dövmeyen dizini döver!
  - İtfaiyeci ve işçi/elektrikçi.
  - Proactive yaklaşımları işleyeceğiz.
- Risk tanımı:
  - Tam bir uzlaşısı yok.
  - Uzlaşılan özellikler:
    - Olasılık: Belirli bir risk ortaya çıkabilir veya çıkmayabilir, risk %100 olasılıkla ortaya çıkacak diye bir şey yoktur.
    - Kayıp: Risk ortaya çıktığında istenmeyen sonuçlar ve kayıplar doğurur.
- Genel risk çeşitleri:
  - Proje riskleri
  - Teknik riskler
  - İş riskleri
- Farklı risk sınıflandırmaları ve risk işleme önerileri için bkz. SEI, ISO, ANSI, makaleler, kitaplar, vb.

141

## YAZILIM PROJE YÖNETİMİ

### RİSK YÖNETİMİ

- Proje riskleri:
  - Proje planını tehdit eder.
  - Gerçekleşirse zamanlama ileri tarihlere sarkar ve maliyet artar.
  - Örnekler: Bütçe riskleri, Zaman riskleri, Personel riskleri, vb.
- Teknik riskler:
  - Üretilen yazılımın kalitesini ve zamanında bitirilmesini etkiler.
  - Gerçekleşirse yazılımı gerçekleştirme zorlaşır veya imkansızlaşır.
  - Çözümleme, tasarım, gerçekleştirme ve bakım aşamaları ile ilgili risklerdir.
  - Örnek: 'Son teknoloji' ürünler yüksek teknik riske sahiptir.
    - Bug'lar var mı? Dokümantasyonu tam mı? Tam anlayabildik mi bu teknolojiyi? Yarın da bu teknoloji hayatta olacak mı?
- İşletme riskleri:
  - Pazar riskleri: Ürüne talep olur mu? Ör. Tıraş bıçağı, tıraş makinesi
  - Satış riskleri: Pazarlama ekibi ürünü nasıl satacağını biliyor mu?
  - MIS/MBA konuları!

142

## YAZILIM PROJE YÖNETİMİ

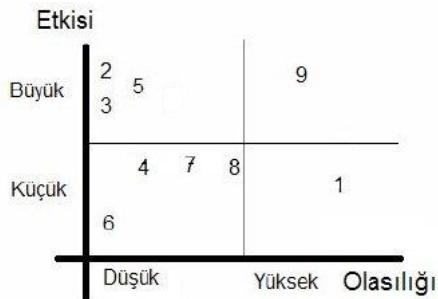
### RİSK İŞLEME VE RİSK TABLOSU

- Risk tablosu oluşturma:
  - Tüm ekip veya risk işleme ekibinin tüm üyeleri olası riskleri adlandırsın.
  - Risklerin çeşitlerini belirleyin (proje, teknik, iş)
    - Herkes kendi uzmanı olduğu risk alanında riskleri alt türlere ayırsın.
    - Ör. Teknik: Planlanandan daha düşük yeniden kullanım, kullanılan gereçlerde deneyimsizlik, vb.
  - Risklerin gerçekleşme olasılıklarını belirleyin.
    - Düşük, orta, yüksek gibi kategoriler.
  - Risklerin etkilerinin büyüklüğünü değerlendirin
    - Küçük bir sıyrık, haydi gayret, siz beni bırakın.
    - küçük kıyamet, büyük kıyamet.
  - Bu bilgilerden bir tablo yapın.
    - ID, ad, çeşit, olasılık, etki.
    - Listeyi bir yerden sonra kesin (Düşük olasılık ve etkileri atın).
    - Ele alacağınız riskler için risk bilgi sayfaları oluşturun.
      - Bu sayfa risk hakkında bilgiler ve önleme yollarını içersin.
      - Önlenemeyen riskler için ise alternatif planlar içersin.

143

### ÖRNEK ÖZET RİSK TABLOSU ve GRAFİĞİ

Risk ID	Adı	Türü,Grubu	Etkisi	Olasılık
01	Organizasyon	Proje	Orta	Yüksek
02	Tahmin	Proje	Büyük	Düşük
03	Kullanışlılık	Teknik	Büyük	Düşük
04	Doğruluk	Teknik	Orta	Düşük
05	Güvenilirlik	Teknik	Büyük	Düşük
06	Personel	Proje	Küçük	Düşük
07	Araçlar	Teknik	Orta	Orta
08	Metot	Teknik	Orta	Orta
09	Pazarlama	İş	Büyük	Yüksek



- Sonuç:
  - 4, 6, 7, 8. risklerin izlenmesine gerek yoktur.
  - Diğer riskler için ayrıntılı risk bilgi sayfaları oluşturulmalıdır.

144



## ÖRNEK RİSK BİLGİ SAYFASI

### RİSK #09: Pazarlama

**Olasılık:** Yüksek    **Etki:** Büyük    **Türü:** Ticari

**Açıklama:** Firmamız sadece yazılım geliştiricilerden oluşan çekirdek kadroyla çalışmaktadır. Ortaya çıkaracağımız yazılımın piyasada çok fazla alternatifi vardır.

#### İşaretleri:

1. Müşteri adayları ile görüşmede dinleyicilerin ilgisizliği ve "Biz zaten X yazılımını kullanıyoruz. Neden sizi seçelim?" türü yorumları.
2. Bakım aşamasının ilk altı aylık süresince ürünümüzün kendisini amorti etme oranının %50'nin altında kalması

#### Önlemler:

1. Bir programcımıza MBA yaptırmak.
2. Gereksinim mühendisliği aşamasında konu uzmanlığı odaklı danışmanlık hizmeti almak.
3. Bir ya da iki firma ile pilot uygulama yapmak ve bu firmalara ürünü özel indirim ile satmak

145

Bu yansı ders notlarının düzeni için boş bırakılmıştır.

146

## YAZILIM MÜHENDİSLİĞİ DERS NOTLARI

Yrd.Doç.Dr. Yunus Emre SELÇUK

### YAZILIM SINAMA TEKNİKLERİ

147

### YAZILIM SINAMA TEKNİKLERİ

#### GENEL BİLGİLER

- Yazılım geliştirme karmaşık bir süreç olduğundan, hataların ortaya çıkması kaçınılmazdır.
- Yazılım, yaşam döngüsünün her aşamasında, hatalara karşı sınanır.
  - Gereksinimler arasındaki tutarsızlıklar,
  - Çözümleme şeması ile uygulama alanı arasındaki uyumsuzluklar,
  - Tasarım hataları,
  - Çalışma anı hataları,
  - vb.
- Yaşam döngüsünde ilerledikçe, hataların düzeltilmesi zorlaşacaktır.
- İyi bir sinama yaklaşımı, hataların erken belirlenmesine katkıda bulunacak ve yazılımın kalitesini arttıracaktır.
- Bir sinama yaklaşımı şu bileşenleri içermelidir:
  - Planlama
  - Tasarım
  - Çalıştırma
  - Bilgi toplama ve değerlendirme

148

## YAZILIM SINAMA TEKNİKLERİ

### GENEL BİLGİLER

- Sınama yaklaşımlarının genel özellikleri:
  - Sınamalar bileşen düzeyinde başlar ve sistem düzeyinde sonlanır.
    - Ürün tek başına bilgisayar yazılımından ibaret olmayabilir.
    - Sistemin bütününde çeşitli algılayıcılar, harici sistemler, vb. yer alabilir.
    - NYP'de sınıfların bireysel sorumluluklarından başlayıp, yazılımın üst düzey sorumluluklarına kadar tüm işlevler sınanır.
  - Yazılım yaşam döngüsünün değişik aşamalarında değişik sınama teknikleri uygun olacaktır.
  - Projenin değişik aşamalarında değişik sınama teknikleri uygun olacaktır.
  - Sınamalar hem ilgili bileşeni oluşturan kişi tarafından, hem de bağımsız kişiler tarafından yapılır.
    - Büyük ölçekli kurumlarda ayrı bir sınama ekibi bulunabilir.
  - Sınama ve hata ayıklama ayrı işlerdir, ancak hata ayıklama her sınama yaklaşımının önemli bir parçasıdır.
  - Resmi teknik değerlendirmeler sayesinde sınama öncesinde de hatalar belirlenebilir.
    - Formal technical reviews, ileride değinilecek.

149

## YAZILIM SINAMA TEKNİKLERİ

### SINAMA AMAÇLI YAPILANMA

- Bir bileşen ilk olarak onu hazırlayan programcı tarafından sınanır.
  - Bir programı en iyi olarak bilen, onu yazandır.
- Her bileşen aynı zamanda yazarı dışındaki kişilerce sınanır.
  - Yazılım evinin bağımsız bir sınama ekibi bulunabilir (ITG: Independent Test Group).
    - Aksi halde farklı projelerde çalışan ekipler, veya aynı projenin farklı alt grupları, birbirlerinin çalışmalarını sınar.
  - Programcıların çıkarları, kendi yazdıkları programın hatasız, zamanında ve bütçe dahilinde tamamlanmasını gerektirir.
  - Psikolojik açıdan bakıldığında sınama 'yıkıcı' bir eylemdir.
  - Bu nedenlerle geliştiriciler kendi programlarının hatalarını bulmaya yönelik değil, doğru çalıştığını ispatlamaya yönelik sınamalara eğilim gösterebilir.
- Yazılım teknik olmayan kişiler tarafından da sınanır (ileride değinilecek).
- Sonuç:
  - Müşteriler hataları önünde sonunda bulacaklardır. İyisi mi siz onlardan önce davranın! Aksi halde prestijiniz sarsılacaktır.

150

## YAZILIM SINAMA TEKNİKLERİ

### SINAMA YAKLAŞIMININ BELİRLENMESİ

- Göz önünde bulundurulması gereken konular:
  - Sınama çalışmaları, en çok çaba gerektiren yazılım mühendisliği etkinlikleri arasında yer almaktadır.
  - Sınama çalışmaları, olası tüm hataların yakalanacağı bir 'güvenlik ağı' olarak düşünülmemelidir.
    - Sınama için ne kadar çaba gösterilirse gösterilsin, tespit edilemeyen hatalar olacaktır.
  - Hedefler açıkça belirlenmelidir:
    - Testlerin kapsama alanı,
    - Sınama çalışmalarına ne kadar kaynak ayrılacağı,
      - zaman, kişi, bütçe
    - Yazılımın türüne göre belirlenecek diğer çeşitli ölçütler
      - İki hata arası ortalama süre (MTBF), hassasiyet, vb.
  - Sınamalar otomatikleştirilmelidir.
  - Sınama süreci ölçülmeli ve iyileştirilmelidir.

151

## YAZILIM SINAMA TEKNİKLERİ

### SINAMA TÜRLERİ

- Yaklaşım tarzlarına göre sınama türleri:
  - Kara kutu sinaması (Black-box testing): Sınanacak birimin iç işleyişi bilinmez, sadece birimin beklenen girdilere karşı beklenen çıktıları üretip üretilmediğine bakılır.
  - Beyaz kutu sinaması (White-box testing): Sınanacak birimin iç işleyişi bilinir ve yapılacak sinamalar buna göre belirlenir.
- Yürütölme sıralarına göre sınama türleri:
  - Doğrulama Sinamaları (verification tests): Yazılım ekibi tarafından yapılır.
    - Birim sinamaları
    - Tümlleştirme sinamaları
  - Geçerleme Sinamaları (validation tests): Son kullanıcılar tarafından yapılır.
    - Alfa sinaması
    - Beta sinaması

152

## YAZILIM SINAMA TEKNİKLERİ

### DOĞRULAMA SINAMALARI

- Birim sınamaları (Unit testing)
  - En küçük yazılım bileşeninin sınanmasıdır.
  - NYP'de bireysel sınıfların sınanmasıdır.
  - Ne zaman tasarlanır?
    - Kodlamadan önce (çevik yaklaşım),
    - kodlama sırasında,
    - veya kodlamanın ardından.
  - Kodlama sırasında veya kodlamanın ardından yürütülebilir.
  - Bir sınıfın tek başına yürütemediği sorumlulukların sınanması için, bu sınıfın ihtiyaç duyduğu diğer sınıfların yerine geçecek kod gerekebilir.
    - Vekil, sahte, yalancı kod/sınıf, vb.
    - Stub, dummy, surrogate, proxy, vb.
    - Vekil sınıflar, sadece ihtiyaç duyulan sınıflar gerçekleştirilene dek kullanılır.
    - Vekil sınıfların basit tutulması, ek kodlama yükünü azaltır.
      - Bu mümkün değilse, ortaklaşa yürütülen sorumlulukların sınanması tümleştirme sınamalarına bırakılır.

153

## YAZILIM SINAMA TEKNİKLERİ

### DOĞRULAMA SINAMALARI

- Birim sınamalarında aranabilecek hata türleri:
  - Farklı veri tiplerinin karşılaştırılması veya birbirinin yerine kullanımı.
    - NYP'de: Çokbüyümlülüğün yan etkileri
  - Mantıksal işleçlerin yanlış kullanımı
  - İşleçlerin önceliklerinin gözden kaçırılması
  - Değişkenlerin karşılaştırılmasındaki hatalar
  - Döngülerin hatalı sonlanması veya sonsuz döngüler
  - Değişkenlere hatalı değerler atanması
  - vb.
- Yaptığınız hatalardan ders çıkarın:
  - Yapılabilecek tüm kodlama hataları öngörülemez, ancak kariyeriniz boyunca her hata yaptığınızda bu hatanızı 'aranabilecek hata türleri' listenize ekleyin.

154

## YAZILIM SINAMA TEKNİKLERİ

### DOĞRULAMA SINAMALARI

- Tümlleştirme sinamaları (Integration testing)
  - Sınıflar birim sinamalarını gemişlerse, bir araya getirildiklerinde de doğru alışmazlar mı?
    - Yazılım geliştirme sürecinin her aşamasında her ayrıntının açıka belirlenmesi beklenemez.
    - Ayrı ayrı programcılar, belirlenmemiş ayrıntılar üzerinde kendi karar verme yetkilerini (initiative) kullanabilir.
    - Aynı kişinin farklı ayrıntılar hakkında verdiği kararlar bile birbiri ile uyumlu olmayabilir.
  - NYP'de birim ve tümlleştirme sinamalarını birbirinden ayıran kesin çizgiler yoktur.

155

## YAZILIM SINAMA TEKNİKLERİ

### DOĞRULAMA SINAMALARI

- Tümlleştirme sinamaları türleri:
  - Tahribat sinaması (smoke testing)
    - Yüzeysel ancak başarısız olma durumunda tüm sistemin alışmasının olanaksız olacağı sinamalardır (show-stopper errors).
    - Diğer tümlleştirme sinamalarından önce yapılır.
    - eşitli bileşenler tüm gerekli yazılım elemanlarını içeren (kod, yapılandırma dosyaları, dış kütüphaneler, vb.) paralar bir araya getirilir (build) ve günlük olarak sinanırlar.
  - Geriye dönük sinama (regression testing)
    - Yazılıma yeni bir işlev veya bileşen eklendiğinde, tüm sinamaların yenilenmesidir.
    - Hangi ölçekte bir eklentinin geriye dönük sinamayı başlatacağının kararını vermek gerekir.
      - Ölek düştüke sinama sıklaşır ve masraf artar.
  - Otomatik sinama gereleri kullanılarak masraflar azaltılabilir.

156

## YAZILIM SINAMA TEKNİKLERİ

### GEÇERLEME SINAMALARI

- Geçerleme sinamaları (validation testing):
  - Gereksinimler belgesinde yazılmış olan işlevsellikten yola çıkılarak, kullanıcı tarafından yapılır.
  - Son kullanıcıların yapabileceği beklenmedik davranışların tümünü, teknik ekip önceden bilemez.
    - 'Fincan tutacağıının çalışmaması', 'Pencereyi açmak', ...
  - Bir şeyi ne kadar çok kişi incelerse, ondaki kusurlar o kadar çabuk bulunur ve düzeltilir.
    - Eric Raymond: "given enough eyeballs, all bugs are shallow"
  - Alfa ve Beta sinaması olmak üzere iki türü vardır.

157

## YAZILIM SINAMA TEKNİKLERİ

### GEÇERLEME SINAMALARI

- Alfa sinaması:
  - Yazılım firması içerisinde, kullanıcı tarafından yapılır.
  - Yazılım geliştirme ekibinin denetiminde ve izlemesi ile yapılır.
  - Yazılımın doğal kullanım ortamına en yakın koşullarda yürütülür.
- Beta sinaması:
  - Müşterinin kendi yeri içerisinde, gerçek kullanım ortamında yapılır.
  - Yazılım geliştirme ekibi müdahil olmaz.
  - Bulunan hatalar yazılım geliştirme ekibine düzenli aralıklarla ve resmi bir biçimde bildirilir.

158

## YAZILIM SINAMA TEKNİKLERİ

### SİSTEM SINAMALARI

- Yazılım tek başına sistemin bütünü oluşturmayabilir.
  - Gömülü uygulamalar, ara katman yazılımları, vb.
- Sistemin tümünü her yönüyle incelemeye yönelik sınamalardır:
  - Kurtarma (recovery) sınaması:
    - Hatalara dayanıklı (fault tolerant) sistemler için geçerlidir.
    - Bir hata ortaya çıktığında sistemin kendini toparlayarak doğru çalışmaya devam edip edemediği sınanır.
    - Kurtarma işlemi belli bir süre içerisinde tamamlanmalıdır (MTTR: Mean Time To Repair).
  - Güvenlik (security) sınaması:
    - Tek kural: Kural yok!
    - Her güvenlik önünde sonunda aşıılır!
  - Zorlama (stress) sınaması:
    - Normalin dışında yüklenme durumunda, sistemin nereye kadar dayanabileceğinin sınanması
  - Başarım (performance) sınaması:
    - Gerçek zamanlı uygulamalarda özel öneme sahiptir.
    - Program kendisinden bekleneni doğru yapabilir ama zamanında yapamayabilir.

159

## YAZILIM SINAMA TEKNİKLERİ

### HATA AYIKLAMA (DEBUGGING)

- Yapılan türlü sınamaların sonucunda bulunan hatalar düzeltilmelidir.
- Sınama ve hata ayıklama çalışmaları kimileri tarafından 'angarya', 'ayak işi', 'sıkıcı', 'ikinci sınıf' olarak nitelendirilebilir.
  - Sınamanın önemini gördük
  - Yazılımındaki hatalar size ve kurumunuza prestij kaybettirir.
  - Düzeltilmeyen veya düzeltilmesi uzun süren hatalar ise daha çok prestij kaybettirir.
  - Sınama ve güvenlik açıklarının belirlenmesi için alışlagelmiş düşünce biçiminin dışına çıkabilmek gerekir ki bu da özel bir yetenektir.
  - Saygı duyulan ve ünlü programcılar, hata ayıklamanın kodlama yapmaktan daha zor ve daha çok yetenek isteyen bir iş olduğu görüşünde birleşmektedirler.
- Bir noktada takıldığınızda, biraz ara verip sorunu yeniden incelemeniz, başarı şansınızı arttıracaktır.
- Geliştirme ortamının hata ayıklama yeteneklerinden sonuna kadar yararlanın.

160



## YAZILIM SINAMA TEKNİKLERİ

### YAZILIM GÖZDEN GEÇİRME EYLEMİ

- Yazılım gözden geçirme: Software review
  - Yazılım sınavasından önce, sınama eylemlerinden daha az masrafla, yazılım hatalarının bulunmasını amaçlar.
  - Sınama çalışmaları ile bulunabilecek tüm hatalar gözden geçirme ile bulunamaz, ancak gözden geçirme daha verimlidir.
  - Gözden geçirme resmi veya gayri resmi olabilir.
    - Çalışmalar resmi gözden geçirmelerin daha etkili olduğunu göstermiştir.
    - Gözden geçirme toplantılarının yapılması durumunda.
    - Çevik süreçlerden XP'deki eşli programlama da bir tür gözden geçirir.
- Resmi yazılım gözden geçirme: Formal software review
  - Toplantı şeklinde yapılır.
  - Toplantının bir yöneticisi (review leader) bulunur.
  - Bu konuda çeşitli standartlar önerilmiştir.
    - Ör: IEEE 1028 standardı

161

## YAZILIM SINAMA TEKNİKLERİ

### RESMİ YAZILIM GÖZDEN GEÇİRME ÇALIŞMALARI

- IEEE 1028 std. göre bir resmi yazılım gözden geçirme adımları:
  - Değerlendirme başlangıcı: Değerlendirme yöneticisi standart bir checklist kullanarak, verimli bir toplantı için gerekli koşulları sağlar.
  - Yönetimin hazırlanması: Sorumlu yönetim gözden geçirme için gerekli kaynakları hazırlar ve toplantının standartlara uygun yürütülmesini sağlar.
  - Gözden geçirme prosedürlerine genel bakış: Değerlendirme yöneticisi tüm değerlendiricilerin gözden geçirmenin amaçlarını ve prosedürlerini anladığından emin olur.
  - Bireysel hazırlık: Değerlendiriciler bireysel olarak inceleme toplantısına hazırlanır. Bu amaçla gözden geçirilecek malzemedeki hataya yol açabilecek olası bozukluklar (anomaly) aranır.
  - Grup incelemesi: Bireysel hazırlıkların sonuçları önceden belirlenen yer ve zamandaki toplantıda bir araya getirilir ve sonuç raporu üzerinde uzlaşıya varılır.
  - Düzenleme: İncelenen çalışmanın yazar(lar)ı veya atanacak bir başka kişi/ekip, önceki adımda belirlenen noktaları düzeltir.
  - Sonlandırma: Değerlendirme yöneticisi düzeltmelerin yeterliliğini inceler.

162