**BLM 426 YAZILIM MÜHENDİSLİĞİ**

**BAHAR 2016**

**Yrd. Doç. Dr. Nesrin AYDIN ATASOY**

**5. HAFTA: YAZILIM İSTERLERİ ÇÖZÜMLEMESİ**

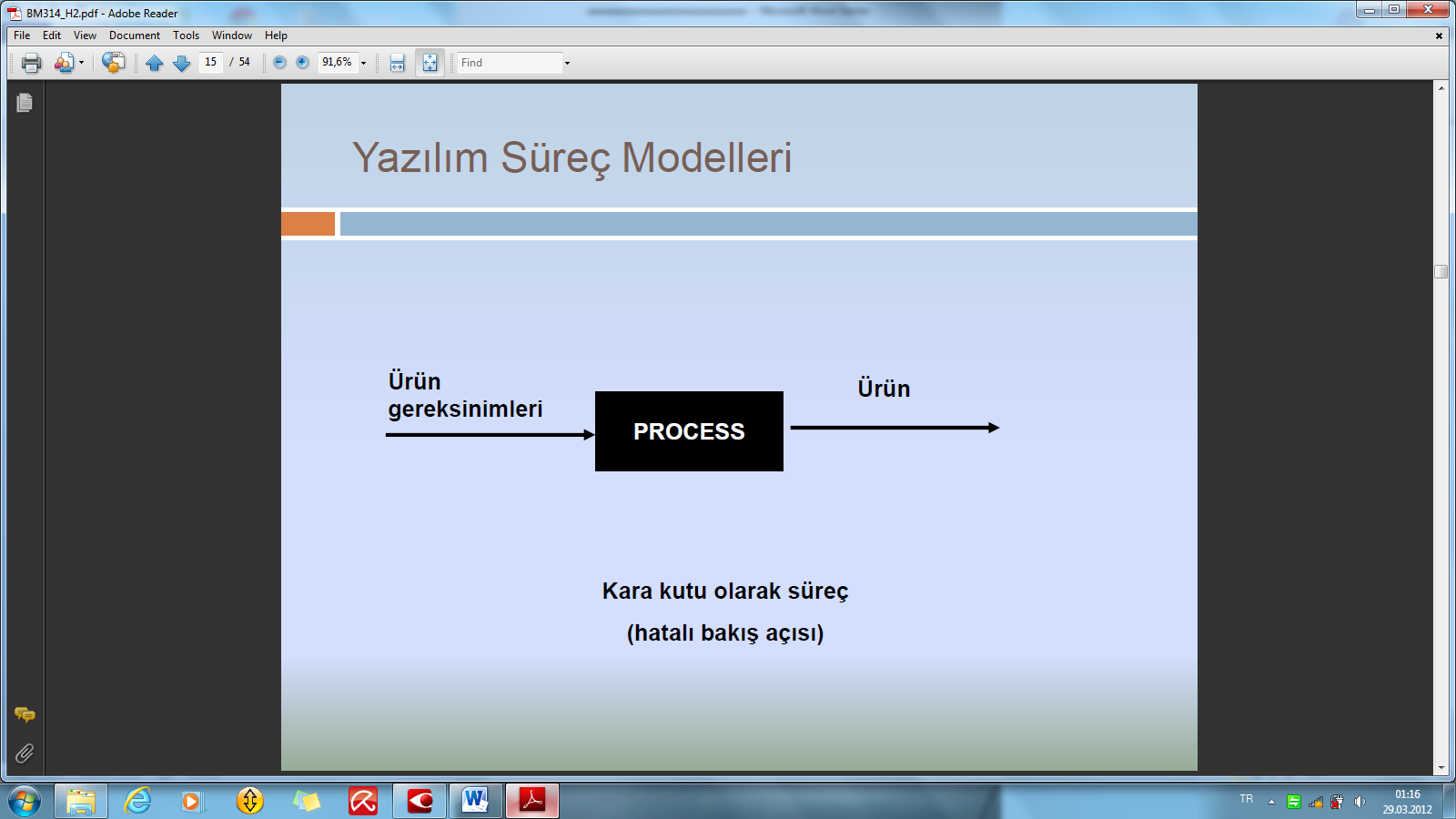
* **Çözümleme Yöntemleri**
* **Nesneye Yönelik Çözümleme**

**Çözümleme:** 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.**

Nesneye yönelimin çözümlemeye uygulanması günümüzde kullanılan en yaygın tekniklerden biridir. Klasik çözümlemelerde yer alan **giriş, işleme ve çıkış şeklinde kullanılan bilgi akışı yerine, nesneye yönelik çözümleme, nesnelere, onların özelliklerine ve onlar üzerinde yapılan işlemlere yoğunlaşır.**



* **“Nesne” Nedir?**

İyi tanımlı bir kapsamı ve kimliği olan, belirli bir durum ve davranışı içeren, soyut veya somut varlıktır.

* Nesne gerçek dünyadaki somut bir varlığı temsil edebilir.

Televizyon, motor, vb.

* Nesne tamamen kavramsal bir varlığı temsil edebilir.

Banka hesabı, vb.

**Not:** ymg\_hafta\_5\_ekle\_1 ‘ e git (1-14 sf anlat).

* **Nesne Tanımlama**

Nesne tanımlamayı kolaylaştırabilmek için nesnelerin alabileceği biçimlere göz atalım;

* Sistemin dışında yer alan ve sistemin kullanacağı bilgiyi üreten ya da tüketen unsurlar birer dış varlıktır (giriş/çıkış aygıtları, işletmenler, başka sistemler gibi) ve nesne olarak tanımlanabilir.
* Sistemin çalışması sırasında meydana gelen ara-işlem, belirli bir hareket, bir aktarım ve bunların tekrarı gibi olaylar da birer nesne olabilirler.
* Sistemle etkileşimde bulunan insanların görevleri ya da rolleri nesne olabilirler (satıcı, alıcı, işçi, öğretmen, yönetici gibi).
* Belirli bir uygulamaya ait grup, ekip, bölüm gibi örgüt birimleri birer nesnedir.
* Uygulama alanı içinde garaj, apartman, okul, üretim hattı gibi yer belirten adlar birer nesnedirler.
* Varlıkların sınıflandırılmasında kullanılan çeşitli yapılar, araç ve gereçler (taşıtlar, bitkiler, kitaplar) birer nesne olarak tanımlanabilir.
* Bu biçimler akılda tutularak, önce problem alanındaki nesne olma olasılığı bulunan tüm isimler çıkartılır. Sonra bunların içinden çözümlemeye dahil edilecek nesne aday­ları saptanır. Aday nesneler içinden aşağıdakiler asil nesne olarak belirlenir:
* Sistemin etkileşimde bulunduğu dış varlıklar
* Sistemin doğru işleyebilmesi için gerekli olanlar
* Özniteliklerinin değiştirilebilmesine olanak sağlayan yordamlara gereksinim duyanlar
* Belirli bir tipten üretilebilenler
* Ortak özellikleri ve ortak yordamları (yani metotları) olanlar.
* **Nesne Sınıflandırma**

Nesneler arasında ilişki kurmak, ortak özelliklerini belirlemek ve tekrar kullanabilmek için nesnelerin sınıflandırılması faydalıdır. Kullanım yerleri ve amaçlarına göre nesneleri şöyle sınıflandırabiliriz:

* Denetim
* Uygulama
* Veri yönetimi
* Arayüz
* **Modelleme**

Nesnelerin belirlemesinden sonra nesneye yönelik çözümlemenin modellenmesine başlanır. Çok çeşitli modelleme yöntemleri ve gösterim şekilleri vardır. Modelleme, nesnelerin, yapılarını ve aralarındaki ilişkilerin şekilsel olarak gösterilmesi temeline dayanır.

Büyükçe bir yazılımın çözümleme modelinde yüzlerce nesnenin yer aldığı yapılar bulunabilir. Birbiriyle ilişkili nesne ve yapıları gruplayarak daha büyük nesneler oluşturmak ve bu şekilde üst düzey bir kapsama sağlamak da mümkündür.

Nesnelerin yaratımları (instance) arasındaki ilişkiler ve ileti yolları da bu modelde yer alır. Hangi nesnenin kime, ne kadar ileti göndereceği, ne tür bağlantıları bulunduğu gösterilir.

Yapısal çözümlemede olduğu gibi, bu modellemede kullanılan diyagramlar Yazılım İsterleri Belirtimi için çok önemli ve kullanışlı girdiyi oluştururlar. Bu amaçla, her bir nesnenin belirtimi ayrı ayrı yapılmalı ve aşağıdaki bilgiler kapsanmalıdır:

**1.** Nesnenin adı

**2.** Öznitelik tanımlamaları

a. Öznitelik adı

**b.** Öznitelik içeriği

**c.** Öznitelik tipi

**3.** Yordamlar

**a.** Yordamın adı

**b.** Yordamın arayüz belirtimi

**c.** Bilgi işleme ve hesaplama belirtimi

**d.** Başarım gereksinimi

**e.** Kısıtlar (zaman, özkaynak, erişim)

**4.** Nesnenin girdileri

**5.** Nesnenin çıktıları

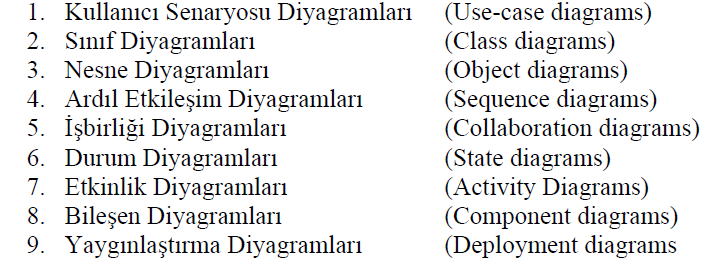
6. Yaratımlar arası bağlantılar

7. İleti bağlantıları

* **Unified Modeling Language (UML)**

UML diyagramı, yazılım ürünlerinin gösterimi için yapı taşları ve ilişkileri gösterir. Fakat;

* Sisteminin nasıl geliştirilmesi gerektiğini tanımlamaz.
* Nesne yönelimli yazılım modellemesi için yapılar sunar, ancak;
  + Bu yapıların hangi sıra ile kullanılması gerektiğini tanımlamaz.
  + Yapıların geliştirme sürecinin hangi aşamalarında kullanılması gerektiğini tanımlamaz.
* **Nesneye Yönelik Tasarım İçin Kullanılan Başlıca UML Elemanları**



* **Use case diyagramı:**       Aktörler ve use case’ler arasındaki ilişkiyi gösterir.
* **Etkinlik (Activity) diyagram:**        Çoğu durumun eylem durumu olduğu ve geçişlerin bir durumdaki eylemin sonuçlanması ile tetiklendiği özel bir durum diyagramı türüdür. Bu diyagram daha çok iç işlemler esnasındaki akışı gösterir.
* **Sınıf diyagramı (class diagram) :** Sistemi oluşturan sınıflar ve bunlar arasındaki ilişkileri gösterir.
* **Ardıl-işlem diyagramı (sequence diagram) :** Nesneler arası etkileşim (davranış),Nesneler arası kontrolün akısı (zamana göre sıralı) gösterir.
* **İşbirliği diyagramı (collaboration diagram) :**Nesneler arası etkileşim (davranış),Nesneler arası kontrolün akısı (mesaj sıralı) gösterir.
* **Durum diyagramı (state diagram):** Dinamik davranıs, olay (“event”) esaslı, Bir nesnenin iç davranısı (nesne yasam döngüsü)
* **Etkinlik diyagramı (activity diagram):**Operasyon akısları
* **Bilesen diyagramı (component diagram):**Uygulamanın fiziksel yapısı, Gerçeklestirme asamasında
* **Yayılma diyagramı (deployment diagram):**Donanım topolojisi,Tasarım ve gerçeklestirme asamalarında
* **KULLANIM SENARYOLARININ GRAFİK GÖSTERİMİ**
* Kullanım senaryoları, ayrıntılı ve uzun belgelerdir.
* Yazılımın yapacağı işlerin özet gösterimi için kullanım senaryosu şemaları kullanılır (use-case diagrams).
* Çizim kurallarını verdikten sonra örnek senaryonun şemasını çiz.
* **Use Case Elemanları**



* **Çizim Bilgileri**

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

**Örnek:**



* **Gereksinim Analizinde Use Case**
* Bakış açısı: Sistem, kullanıcısı için “ne” yapacak ?
  + Sistem kapalı bir kutu (“black-box”)
  + Sistem-kullanıcı etkileşimi
  + Sistemin dışarıdan görünen davranışı
* İlgilenmediklerimiz:
  + Sistemin iç yapısı
  + Sistem belirlenen davranışı “nasıl” yapacak ?
  + Belirlenen davranış “nasıl” kodlanacak ?
* Bu bakış açısı, sistemdeki tüm islevselligi degil, kullanıcılar için artı deger olusturacak işlemleri düşünmemizi sağlar.
* Kullanıcının gereksinimi olmayan özellikleri tanımlamamızı engeller.
* Kullanıcının da anlayabileceği şekilde sistemin davranışlarını ve sorumluluklarını tanımlar.
* Kullanıcı ile iletişimi kolaylaştırır.
* Kullanıcı ara yüzlerinin tasarlanmasını kolaylaştırır.
* Kullanıcı kılavuzlarını yazarken başlangıç noktasını oluşturur.
* Geliştirme sürecini başlatır ve tüm temel is adımlarını birbirine bağlar
* Tasarlanacak test durumlarına esas olusturur
* **Veri Yapılarına Yönelik Çözümleme**

Uygulama yazılımlarında kullanılan bilginin içeriği, akışı ve yapısı vardır. Birçok yazılımında da, genel bilgi işlemenin yanı sıra, sıradüzensel bilgiler, yani veri yapıları bulunmaktadır. Genellikle, giriş akışı ile toplanan veriler işlenerek bilgi depolarına konur ve buradan belirli yapılarda ve akış şeklinde dışarı çıkar. İş dünyası uygulama­ları, giriş dosyalarına göre raporlar üreten çeşitli sistemler, işletim sistemi yazılımları, bilgisayar destekli tasarım ve üretim sistemleri veri yapılarını ağırlıklı olarak kul­landıkları için böyle sistemlerin çözümleme ve tasarımında veri yapılarına yönelik yöntemin uygulanması yararlı olabilir. Günümüzde fazla kullanılmayan ancak çeşitli uygulama şekilleri olan bu yöntemde, ortak olarak, varlık ya da öğe adı verilen bilgi nesneleri, onlar üzerinde yapılan işlemler ve yapılar tanımlanır, mimari ve yordamsal tasarım için altyapı hazırlanır. Bu yönteme uygun olarak sistem geliştirme amacıyla ortaya atılan çeşitli uygulama şekilleri bulunmaktadır. Bunların herbirinin çeşitli gösterim şekilleri ve kuralları vardır, ancak genelde birbirlerine yakındırlar. Biz belirli bir yönteme ağırlık vermeden hepsinde ortak olarak bulunan noktalara göre bu yöntemi kısaca inceleyeceğiz.

Veri yapılarına yönelik çözümlemede öncelikle uygulama alanı bağlamında bilgiyi üreten ve tüketen unsurlar incelenir, bunlardan yazılım sistemi bağlamı içinde yer alabi­lecekler ayrılır. Sonra da uygulamanın işlevleri ortaya konur ve sonuçlar modellenir.

Uygulama alanı bağlamını belirlemek için üzerinde işlem yapılacak bilgi öğeleri ile bunların üreticileri ve tüketicileri tanımlanır. Bu amaçla, veri akış diyagramlarına çok benzeyen varlık diyagramları (entity diagram) kullanılır. Bu diyagramda yer alan yuvarlaklar bilgi üreticilerini ve tüketicilerini, oklar da akan bilgiyi gösterir.

Kullanılan varlıkların sıradüzensel yapısı, işlevler, ara-işlem sırası, zamanlama özel­likleri birer diyagramla gösterilir. Bundan sonra da tasarım aşamasına geçilir.

* **Belgelendirme**

Yazılımın amacına ulaşarak başarılı olabilmesi için iyi bir belirtime dayanması gerek­lidir. Belirtim, isterlerin iyi bir şekilde ifade edilerek belgelendirilmesi işlemidir.

Yazılım belirtimi çeşitli şekillerde hazırlanabilir. Bu bir sunu olabileceği gibi büyük bir belge de olabilir. Çeşitli standartlarda yer almakta olan bu belgeye Yazılım İsterleri Belirtimi (Software Requirement Specifications) adı verilmektedir. Yazılım İsterleri Belirtimi küçük sistemler için bir tane olabilir. Öğelere ayrıştırılabilen büyük sistemlerde ise her öğe için ayrı birer belirtim belgesi hazırlanması daha uygun olur.

Yazılım İsterleri Belirtimi, varsa prototip, yazılım geliştirici ve müşteri tarafından gözden geçirilir. Bu belirtim yazılım geliştirmede kullanılacağı için ayrıntılara, tanımlara, hatta sözcük seçimine çok dikkat edilmelidir. Bu gözden geçirme işlemi önce genel bir bakış açısıyla, daha sonra da ayrıntılı olarak yapılır.

* **Belge İçeriği**

Bir Yazılım İsterleri Belirtimi belgesi içinde bu bilgiler yer almalıdır; Sistemin veya öğenin genel anlatımı, Gerekli sistem durumları ve kipleri, Sistemin veya öğenin herbir işlevinin alt işlevler halinde tanımı, Dışsal arayüz isterleri (herbir arayüz için ayrıntılı halde), İçsel arayüz isterleri, Veri ve denetim akışı, İşlevsel ayrıştırma ve işlevsel tanımlam, İçsel veri isterleri, Uyarlama isterleri, Emniyet isterleri, Güvenlik isterleri, Bilgisayar ortamı isterleri, Bilgisayar kaynakları isterleri (donanım, özkaynak, yazılım, iletişim), Yazılım nitelik etmenleri, Tasarım ve gerçekleştirin! Kısıtları, Personel ile ilgili isterler, Lojistik isterler (bakım, taşıma ortamı), Herbir isterin nasıl sınanacağı ve geçerleme kıstasları

* **Belge Hazırlama Önerileri**

Belirtim belgesi hazırlamada dikkate alınması gerekli bazı noktalar şunlardır:

* Belirtim yalnızca işlevleri içermeli, gerçekleştirimin nasıl yapılacağına değinmemelidir.
* Dış dünyadan gelen etkilere yanıt verme gibi dinamik davranışlara sahip sistemleri tanımlayabilmek için bir belirtim ve modelleme dili kullanılmalıdır.
* Belirtim, sistemi kullanacakların anlayacağı şekilde yapılmalı, tüm belge, açık ve kesin ifadelerle yazılmış olmalıdır.
* Yazılım için belirtilen amaçlar sistemin amaçları ile tutarlı bir şekilde bağlantılı olmalıdır.
* Önemli arayüzler, önemli işlevler gruplandırılmış bir şekilde tanımlanmış olmalıdır.
* Bilgi akışı ve yapıları uygulama alanına uygun şekilde anlatılmalıdır.

**KAYNAKLAR**

1. Yazılım Mühendisliği Ders Notları; Yrd. Doç. Dr. Buket Doğan.
2. Yazılım Mühendisliği; M. Erhan Sarıdoğan
3. Yazılım Mühendisliği; Ali Arifoğlu, Ali Doğru.
4. BBS-651 Yazılım Mühendisliği Ders Notları; A. Tarhan, 2010.