-152116025-
TASARIM SÜREÇLERİ

Ders 9: Tasarım Evresi-4

Dr. Yıldıray Anagün,
yanagun@ogu.edu.tr
Eskişehir Osmangazi University
Computer Engineering Department

Tasarım Evresinde Neler Yapılır?

- Kullanıcı-Sistem ara yüzü tasarımı
- Veri tabanı tasarımı
- Gömülü Donanım tasarımı
- Yazılım tasarımı
- Sistem kontrolleri/test tasarımı.
- Ağ tasarımı ve güvenliği

Yazılım Geliştirmenin Özellikleri

- Günümüz yazılımları, en az diğer mühendislik ürünleri(bina köprü, taşıt, v.b) kadar karmaşıktır.

- Günümüz modern yazılımları, çok sayıda kişinin yer aldığı takımlar halinde yazılıyor. Yazılımlar birçok modülden (sınıf, nesne, hizmet) oluşuyor. Bu da iyi bir iletişim ve modüller arası uyum gerektiriyor.

Yazılım Geliştirmenin Özellikleri

- Yazılımlar sürekli gelişirler. Bu gelişim ve değişim hem yazılım geliştirilirken hem de geliştirildikten sonra olur. Bu nedenle, yazılımlar esnek olmalıdır.
- Bir programlama dilini iyi bilmek kaliteli yazılım geliştirmek için yeterli değildir.
- İyi bir yazılım geliştirmek için, uygun yazılım geliştirme tekniklerini bilmek ve uygulamak gerekir.

Yazılım Geliştirmede Sorunlar

- Yazılımın zamanında tamamlanaması
- Bütçenin aşılması, bakım maliyetlerinin yüksek olması
- Birçok hata çıkması ve hataların giderilememesi
- Yazılımın yeni gereksinimlere göre uyarlanamaması,
- Eski projelerde geliştirilen yazılım modüllerinin yeni projelerde kullanılamaması

Yazılım Geliştirmede Temel Amaç

(1)

(2)

Temel amaç karmaşıklığı çözebilmek, değişimlerle baş edebilmek, ***esnek*** ve ***tekrar kullanılabilir*** yazılım sistemleri geliştirerek, maliyetleri düşürmek.

Yazılım Dünyasındaki Problemler

- Yazılım Geliştirme Maliyetleri
  - İsteklerin çözümlenmesi %3
  - Tasarım %8
  - Kodlama (Programlama) %7
  - Sınama %15
  - Bakım %67

  - Hataların %85'i isteklerin çözümlenmesi ve tasarım aşamalarında oluşmaktadır.

Yazılım Dünyasındaki Problemler

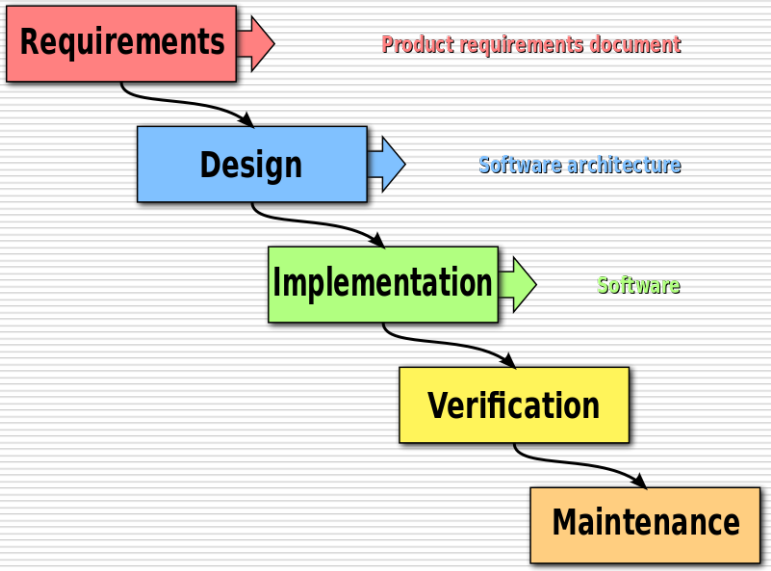
- Hataları Giderme Maliyetleri
  - İsteklerin çözümlenmesi 1 birim
  - Tasarım 1.5-2 birim
  - Kodlama (Programlama) 5-10 birim
  - Sınama 10-15 birim
  - Bakım 15-100 birim

Çözüm

- Yazılım geliştirme hem bir bilim dalı hem de bir sanat
- Kolay ve kesin bir reçete yok. Sezgiler ve deneyim önemli.
- Aşağıdaki unsurlar doğru şekilde kullanıldığında işler kolaylaşır, başarı olasılığı artar.

Çözüm

- Uygun yazılım geliştirme süreçleri
  - Şelale (Waterfall) Geliştirme: Ardışık geliştirme yöntemidir. Fazlar yukarıdan aşağıya doğrudur.

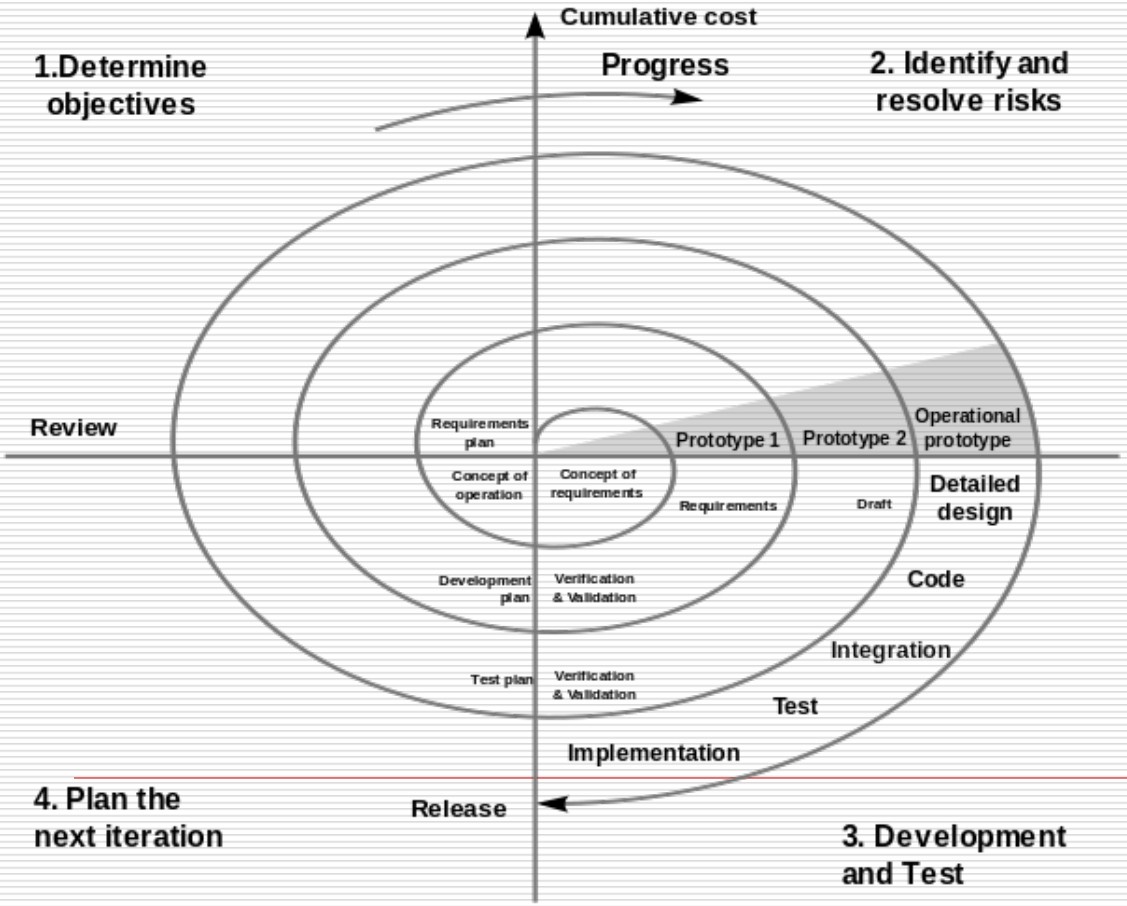


Çözüm

- Prototip Oluşturma (Prototyping): Bu yöntemde prototip yaratmayı kapsamaktadır.

Çözüm

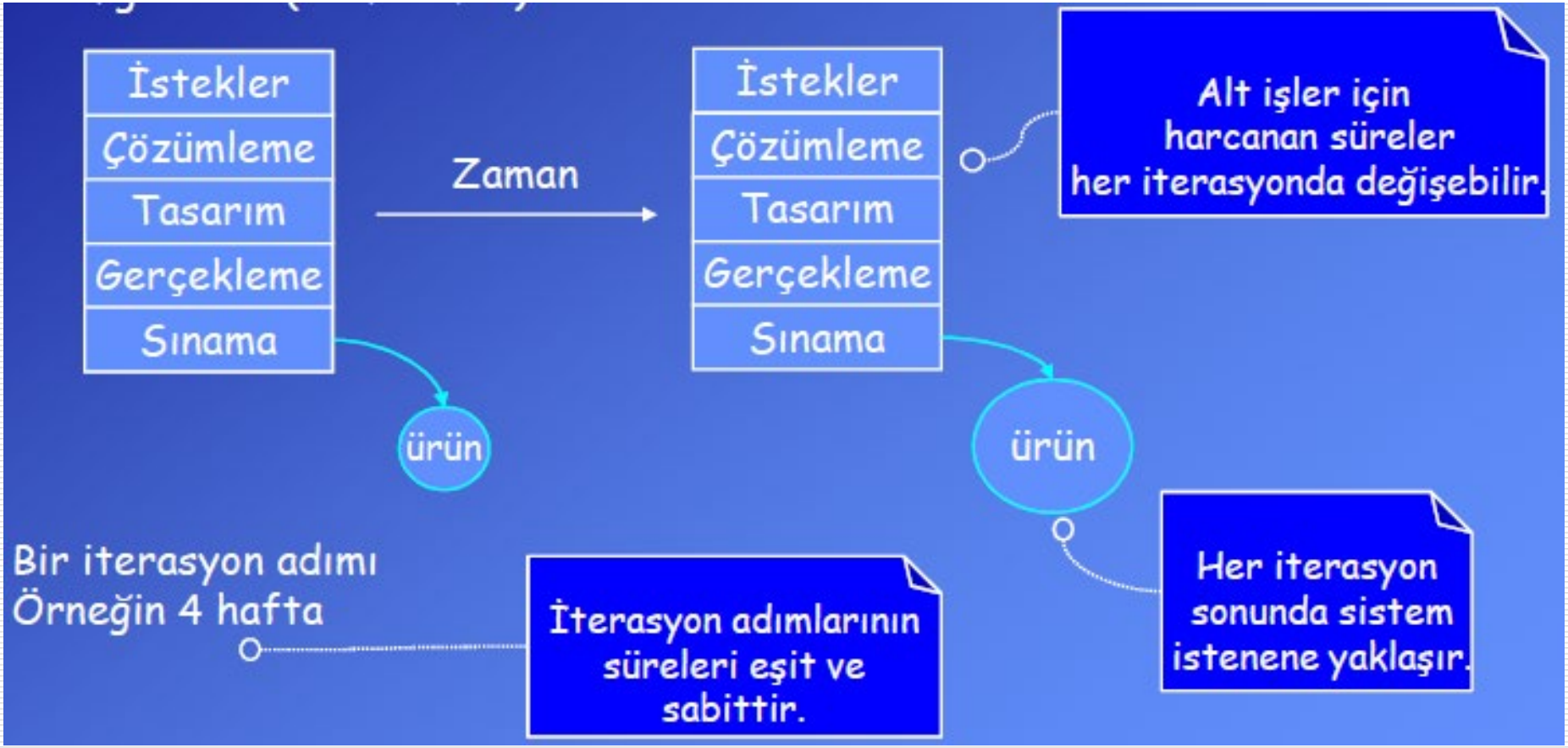
- Spiral Geliştirme:



Çözüm

- Tümleşik Yazılım Geliştirme Süreci (The Unified Process-UP)
  - Yinelemeli (iterative) ve evrimsel (evaluationary) yöntemler
    - Seri halde mini şelaleler gerçekleştirmeyi kapsar. Proje, küçük segmentlere bölünerek geliştirilir, proje geliştirme sürecinde kolay değişiklik imkanı sunar ve proje riski azaltılır.

Çözüm


İstekler
Çözümleme
Tasarım
Gerçekleme
Sınama
Zaman
İstekler
Çözümleme
Tasarım
Gerçekleme
Sınama
Alt işler için harcanan süreler her iterasyonda değişebilir.
ürün
ürün
Bir iterasyon adımı
Örneğin 4 hafta
İterasyon adımlarının süreleri eşit ve sabittir.
Her iterasyon sonunda sistem istenene yaklaşır.

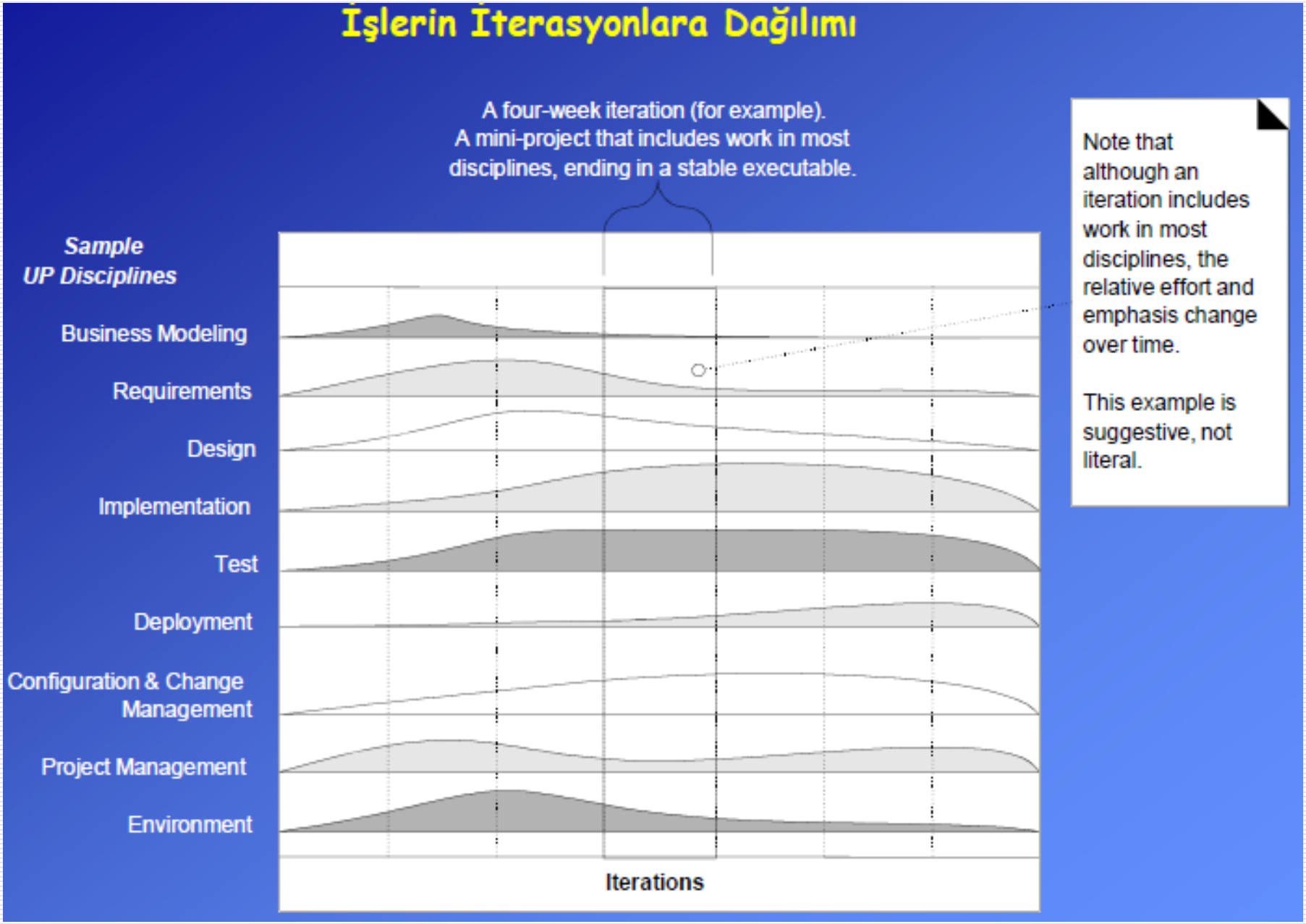
Çözüm

- Yinelemeli Sürecin Yararları
  - Değişen isteklere uyum
  - Erken geri besleme
  - Büyük sistemlerde çözümleme kolaylığı
  - Her iterasyonda deneyim kazanma
  - Risklerin erken giderilmesi
  - Erken ürün elde etme, takımda moral yükselmesi

Çözüm

- Öneriler
  - 2-6 haftalık sabit süreli iterasyonlar uygulanmalı
  - Yüksek risk içeren kısımlar ilk iterasyonda gerçeklemeli
  - Temel oluşturan (çekirdek) yapılar önce gerçeklemeli
  - Sürek kullanıcılardan geri besleme alınmalı
  - Her iterasyon sonrası ürün sınanmalı
  - Kulanım senaryoları (use cases) kullanmalı
  - Görsel modelleme (UML) kullanılmalı

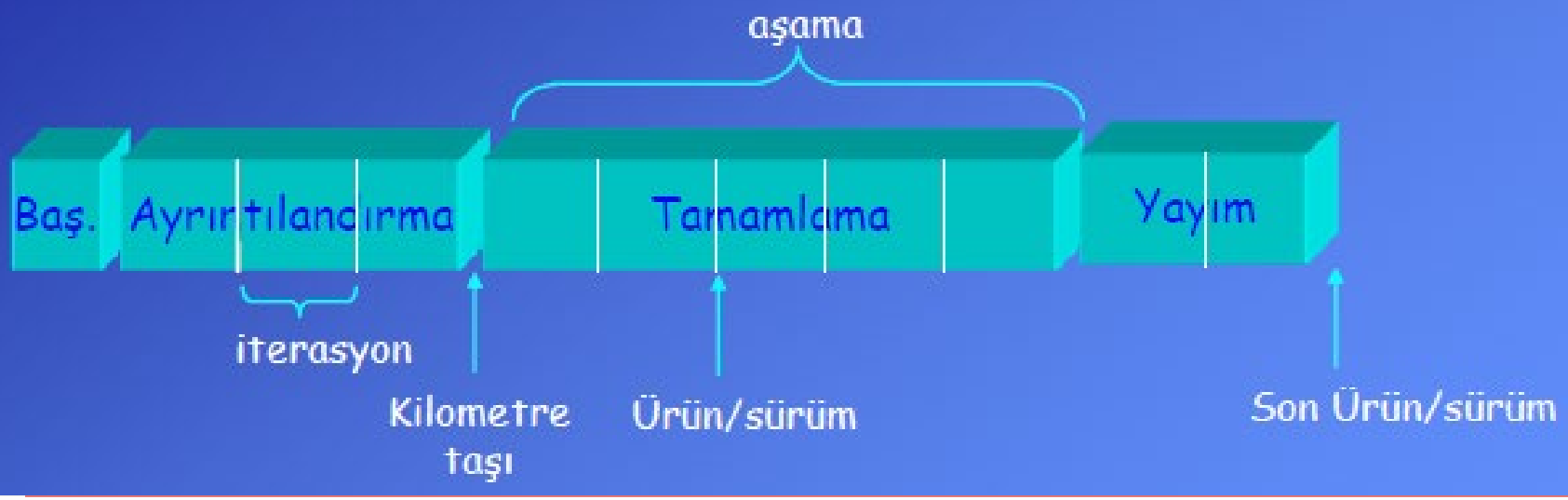
İşlerin İterasyonlara Dağılımı



UP Aşamaları

Tümleştirilmiş Süreçte (UP) yazılım geliştirme aşamaları:

- Başlangıç (*Inception*): Kabaca vizyon, fizibilite, tamam/devam?
- Ayrıntılandırma (*Elaboration*): Daha gerçekçi çözümleme, çekirdek yapının ve yüksek riskli kısımların yinelemeli olarak oluşturulması.
- Tamamlama (*Construction*): Daha az riskli ve düşük öncelikli kısımların yinelemeli olarak gerçeklenmesi.
- Yayım (*Transition*): Beta testleri, piyasaya sürme çalışmaları.



Çözüm

- Programlama ve Modelleme Yöntemleri
  - Nesneye dayalı yöntem

Çözüm

- Yardımcı araçlar
  - UML (The Unified Modeling Language)
  - Yazılım Geliştirme Programları
- Nesneye Dayalı Prensipler

Çözüm: Tasarım Kalıpları (Design Patterns)

- Yazılım sınıflarının belirlenmesinde ve onlara uygun sorumluluklar atanmasında tasarım kalıplarından faydalanılmaktadır.
- Tasarım kalıpları, yazılımda defalarca karşılaşılan bir problemi ve o problemin çözümünde izlenmesi gereken temel yolu tarif etmektedir. Kısaca, aklın yolu birdir.
- Bir problemle karşılaşan tasarımcı, eğer daha önce benzer problemle karşılaşan tasarımcının uyguladığı başarılı çözümü biliyorsa (kalıp), her şeyi yeniden keşfetmek yerine aynı çözümü tekrar uygular.

Çözüm: Tasarım Kalıpları

- Yazılım sınıflarının belirlenmesinde ve onlara uygun sorumluluklar atanmasında tasarım kalıplarından faydalanılmaktadır.
- Tasarım kalıpları, yazılımda defalarca karşılaşılan bir problemi ve o problemin çözümünde izlenmesi gereken temel yolu tarif etmektedir. Kısaca, aklın yolu birdir.
- Bir problemle karşılaşan tasarımcı, eğer daha önce benzer problemle karşılaşan tasarımcının uyguladığı başarılı çözümü biliyorsa (kalıp), her şeyi yeniden keşfetmek yerine aynı çözümü tekrar uygular.

Çözüm

- Tasarım Kalıpları (Design Patterns)
  - Gamma E., Helm R., Johnson R., Vlissides J. adlı dört yazar dörtlü çete (gang of four) olarak adlandırılmışlar ve tasarım kalıplarını bir katalog olarak topladıkları kitabı yazmışlardır.
  - Bu katalog kitapta, tasarım kalıpları bazı bileşenler altında tariflenmiştir.

Tasarım Kalıplarının GoF Sınıflandırması

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method	Adapter (class)	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter (object) Bridge Composite Decorator Flyweight Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

GoF Temel bileşenleri

- Desen Adı
  - Bir model için kısa ve anlamlı bir isme sahip olmak, geliştiriciler arasındaki iletişimi geliştirir
- Sorun
  - Bu kalıbı kullanacağımız sorun ve bağlam nedir?
  - Bu kalıbın kullanılabilmesi için karşılanması gereken koşullar nelerdir?
- Çözüm
  - Tasarım modelini oluşturan öğelerin açıklaması
  - İlişkilerini, sorumluluklarını ve işbirliklerini vurgular
  - Somut bir tasarım veya uygulama değil; daha ziyade soyut bir açıklama
- Sonuçlar
  - Deseni kullanmanın artıları ve eksileri
  - Yeniden kullanılabilirlik, taşınabilirlik ve genişletilebilirlik üzerindeki etkileri içerir

GoF Temel bileşenleri (Devamı)

- Desen Adı ve Sınıflandırması
  - Kalıp ve kalıbın türü için iyi ve öz bir isim
- Amaç
  - Desenin ne yaptığına dair kısa açıklama
- Ayrıca şöyle bilinir
  - Kalıp için diğer isimler
- Motivasyon
  - Modelin nerede yararlı olacağını gösteren bir senaryo
- Uygulanabilirlik
  - Desenin kullanılabileceği durumlar

GoF Temel bileşenleri (Devamı)

- Yapısı
  - Desenin grafik temsili
- Katılımcılar
  - Desene katılan sınıflar ve nesneler
- İşbirlikleri
  - Katılımcılar sorumluluklarını yerine getirmek için nasıl etkileşim kuracaklar?
- Sonuçlar
  - Modeli kullanmanın artıları ve eksileri nelerdir?
- Uygulama
  - Modeli uygulamak için ipuçları ve teknikler

GoF Temel bileşenleri (Devamı)

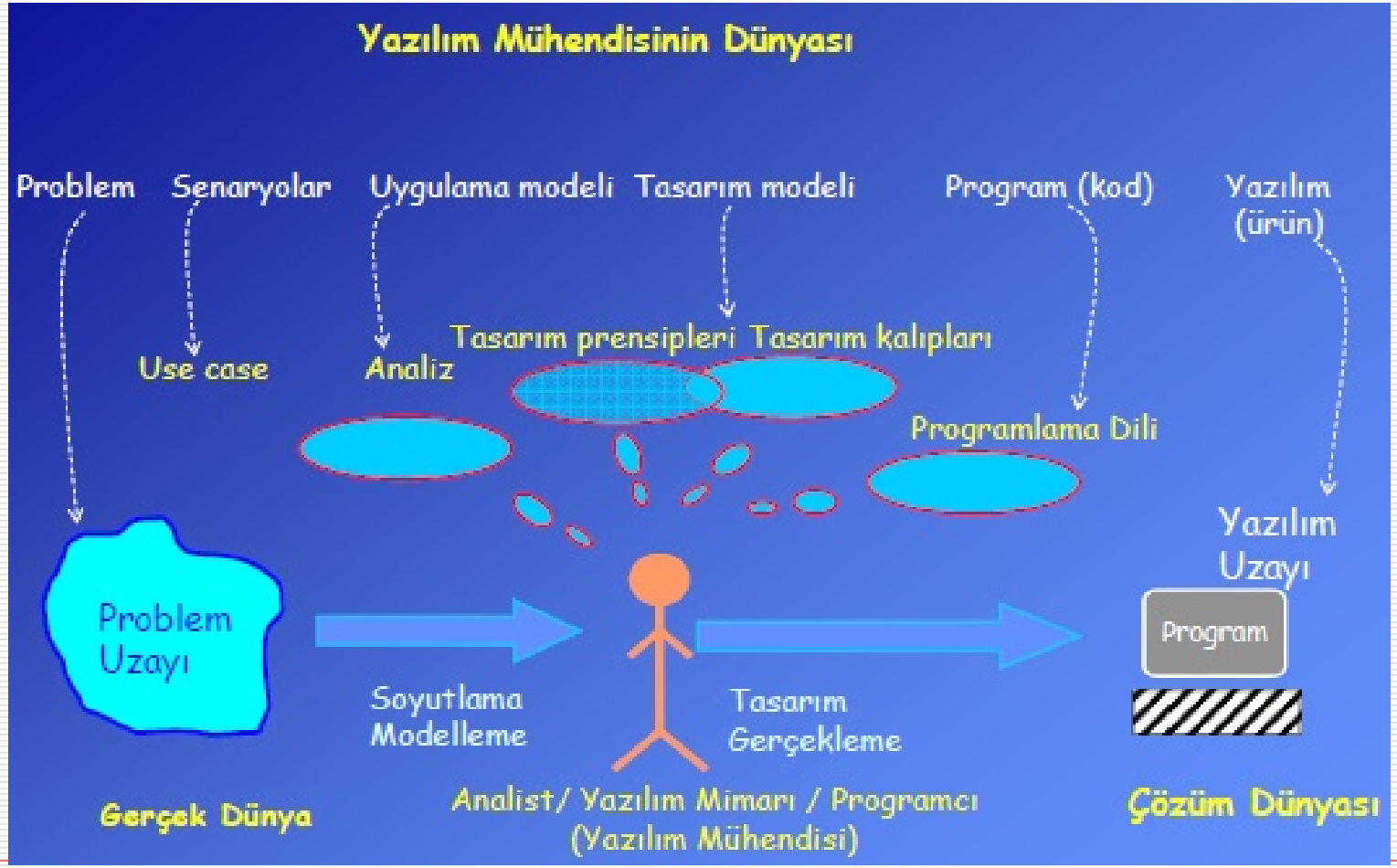
- Basit kod
  - Örnek bir uygulama için kod parçaları
- Bilinen Kullanımlar
  - Gerçek sistemlerde model örnekleri
- İlgili Desenler
  - Desenle yakından ilgili diğer desenler

Yazlım Geliştirme Aşamaları

- İstekler
  - Müşterilerin isteklerinin anlaşılması Kullanım durumlarının (use cases) yazılması
  - Problemin Analizi (Domain Analysis)
    - Sistem ne yapacak?
  - Tasarım (Design)
    - Sistemin işbirliği yapan nesneler şeklinde tasarlanması. Sorumlulukların sınıflara atanması.

Yazlım Geliştirme Aşamaları

- Gerçekleme (Implementation)
  - Kodlama, programlama
- Değerlendirme (Evaluation)
  - Sınama (testing), performans ölçümü ve değerlendirmesi, bakım


Yazılım Mühendisinin Dünyası
Problem
Senaryolar
Uygulama modeli
Tasarım modeli
Program (kod)
Yazılım (ürün)
Use case
Analiz
Tasarım prensipleri
Tasarım kalıpları
Programlama Dili
Problem Uzayı
Yazılım Uzayı
Program
Soyutlama Modelleme
Tasarım Gerçekleme
Gerçek Dünya
Analist/ Yazılım Mimarı / Programcı (Yazılım Mühendisi)
Çözüm Dünyası

Örnek:

- Zar Oyunu: Oyuncu 2 zar atar. Zarların üste gelen sayıları toplamı 7 ise oyuncu kazanır, aksi durumda kaybeder.
- 1. İsteklerin belirlenmesi:
  - Bunun için en uygun yöntem, kullanım senaryoları (use cases) yöntemidir.
  - Bu yöntemde, tasarımı yapılan sistem ile kullanıcılar arasındaki olaylar yazılır.

Örnek:

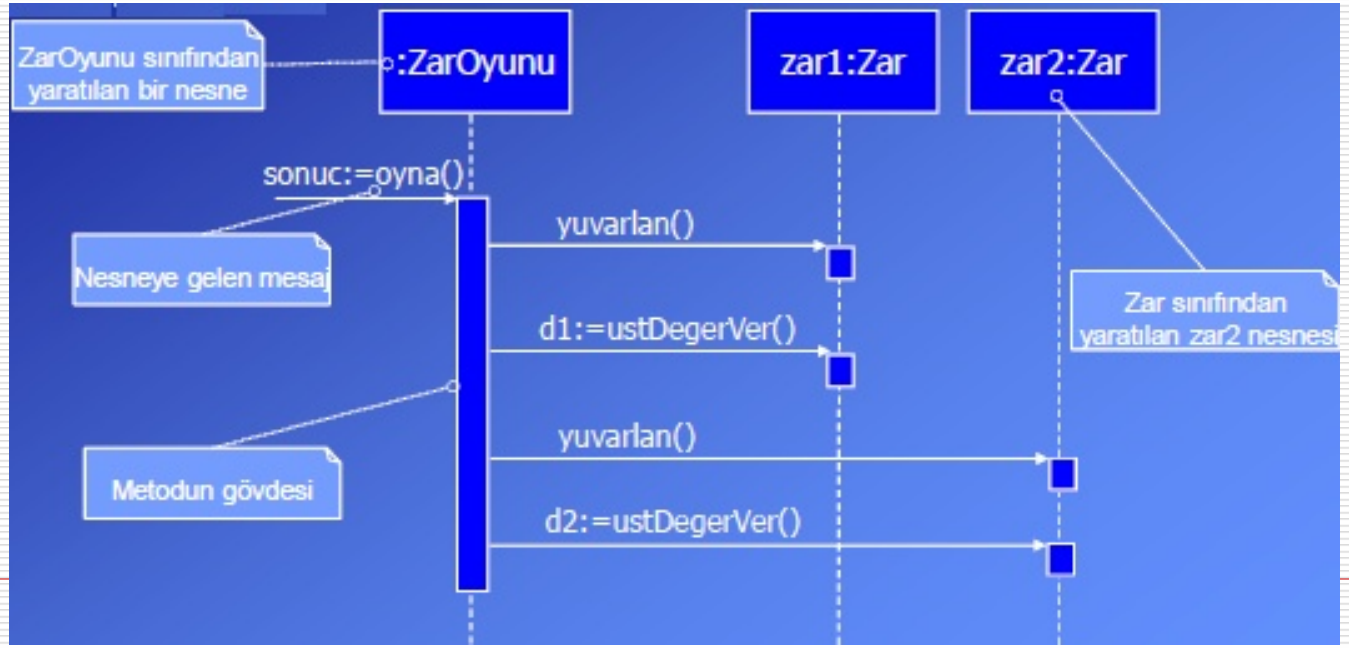
- Ana senaryo:
  - 1. Oyuncu iki zar yuvarlar.
  - 2. Sistem zarların üzerindeki değerleri ve toplamlarını gösterir.
  - 3. Oyun sona erer.
- Alternatif akışlar:
  - 2.a. Üste gelen değerlerin toplamı 7'dir. Sistem oyuncuya kazandığını bildirir.
  - 2.b. Üste gelen değerlerin toplamı 7'den farklıdır. Sistem oyuncuya kaybettiğini bildirir.

Örnek:

- 2. Uygulama Uzayında Modelleme (Analiz):
  - Uygulamayı oluşturan sınıflar ve ilişkileri tanımlanır.

Örnek:

- 3. Yazılım (Çözüm) Uzayında Modelleme (Tasarım):
  - Tasarım yapılarak sistemin sorumlulukları tasarım kalıpları yardımıyla uygun yazılım sınıflarına atanır.
  - **a. Etkileşim Diyagramı:** Nesnelerin davranışları ve aralarındaki etkileşim belirlenir.



Örnek:

- **b. Sınıf Diyagramı:** Sınıfların sahip olacakları üyeler belirlenir.

Örnek:

- 4. Kodlama
- 5. Sınama

- Kaynak
  - Dr. Feza Buzluca ders notları
  - Değişik Tasarım Kalıpları notları