Yazılım Yaşam Döngüsü ve Süreç Modelleri

Bölüm - 2

Doç. Dr. Resul DAŞ

Bu Haftaki Konular

Amaçlar

- Yazılım Yaşam Döngüsü ’nün Projelerde ki Önemini Kavramak?
- Yazılım Yaşam Döngüsünde Önemli Kavramlar
- Yazılım Geliştirmede Süreç Modellerinin Önemi?
- Süreç Modellerinin Gelişimi
- Süreç Modeli Çeşitleri ve Uygulanması
- Süreç Modellerinin Avantajları ve Dezavantajları
- Metodoloji Yaklaşımı
- Örnek bir Metodoloji

Yazılım Yaşam Döngüsü Nedir?

- Yazılım yaşam döngüsü, herhangi bir yazılımın, üretim aşaması ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.

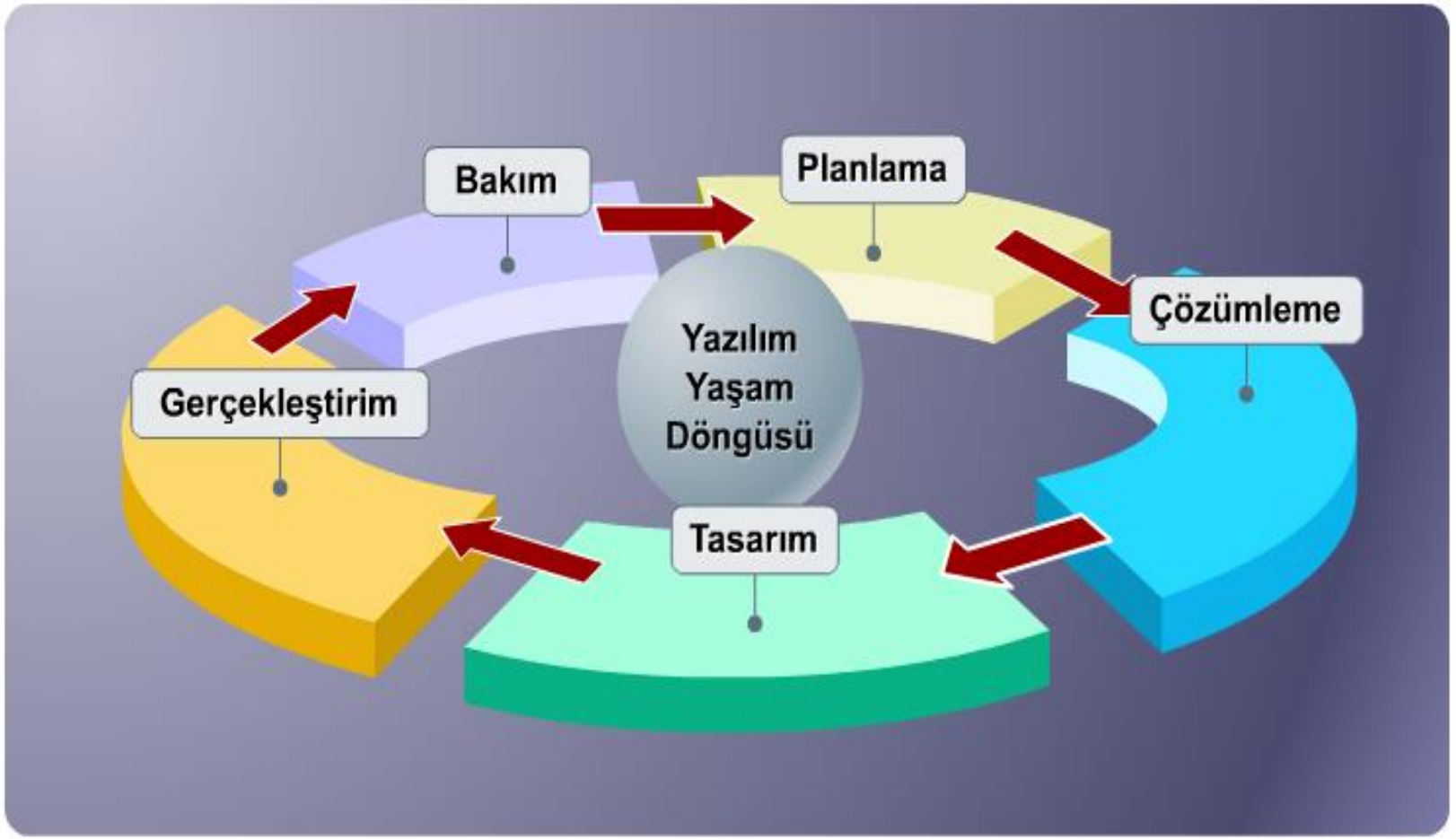
- Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar bir döngü biçiminde ele alınır.

- Döngü içerisinde herhangi bir aşama da geriye dönmek ve tekrar ilerlemek söz konusudur.

- Yazılım yaşam döngüsü tek yönlü ve doğrusal olduğu düşünülmemelidir.

Yazılım Yaşam Döngüsü Temel Adımları

- **Planlama**
- **Çözümleme**
- **Tasarım**
- **Gerçekleştirim**
- **Bakım**



Planlama

- Üretilecek yazılım ile ilgili olarak, personel ve donanım gereksinimlerinin çıkarıldığı, fizibilite çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.

Çözümleme

- Yazılım işlevleri ile gereksinimlerin ayrıntılı olarak çıkarıldığı aşamadır.
- Bu aşamada temel olarak mevcut sistemde var olan işler incelenir, temel sorunlar ortaya çıkarılarak, yazılımın çözümleyebilecekleri vurgulanır.
- Temel amaç, bir yazılım mühendisi gözüyle mevcut yapıdaki işlerin ortaya çıkarılması ve doğru olarak algılanıp algılanmadığının belirlenmesidir.
- Bu aşamada temel UML diyagramlarının çizimine başlanır (Use Case, Activity, Class diagram… vs.)

Tasarım

- Çözümleme aşamasından sonra belirlenen gereksinimlere karşılık verecek yazılım ya da bilgi sisteminin temel yapısının oluşturulması çalışmalarıdır.
- Bu çalışmalar, mantıksal tasarım ve fiziksel tasarım olarak iki gruba ayrılır.
  - **Mantıksal Tasarım:** Mevcut sistem değil önerilen sistemin yapısı anlatılır. Olası örgütsel değişiklikler önerilir.
  - **Fiziksel Tasarım:** Yazılımı içeren bileşenler ve bunların ayrıntıları içerilir.

Gerçekleştirim

- Kodlama
- Test etme
- Kurulum

çalışmalarının yapıldığı aşamadır.

Bakım

- İşletime alınan yazılım ile ilgili olarak, hata giderme ve yeni eklentiler yapma aşamasıdır.

- Bu aşama yazılımın tüm yaşamı boyunca sürer.

Yazılım Yaşam Döngüsü Temel Adımları

- Yazılım yaşam döngüsünün temel adımları çekirdek süreçler (core processes) olarak da adlandırılır.

- Bu süreçlerin gerçekleştirilmesi amacıyla

  - **Belirtim Yöntemleri** (Software Specification Methods) - bir çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemler.

  - **Süreç Modelleri** (Software Process Models) - yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ya da sırada, nasıl uygulanacağını tanımlayan modeller kullanılır.

Belirtim Yöntemleri

- **Süreç Akışı İçin Kullanılan Belirtim Yöntemleri**
  - Süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemler
    (Veri Akış Şemaları, Yapısal Şemalar, Nesne/Sınıf Şemaları).
- **Süreç Tanımlama Yöntemleri**
  - Süreçlerin iç işleyişini göstermek için kullanılan yöntemler
    (Düz Metin, Algoritma, Karar Tabloları, Karar Ağaçları, Anlatım Dili).
- **Veri Tanımlama Yöntemleri**
  - Süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemler
    (Nesne İlişki Modeli, Veri Tabanı Tabloları, Veri Sözlüğü).

Süreç (process) nedir?

- **Süreç** olguların ya da olayların, belli bir taslağa uygun ve belli bir sonuca varacak biçimde düzenlenmesi, ***sıralanması***.

- Bir şeyin yapılışını, üretiliş biçimini oluşturan sürekli işlemler, eylemler dizisi **(Kaynak**: http://tr.wikipedia.org**)**

- Aralarında birlik olan veya belli bir düzen veya zaman içinde tekrarlanan, ilerleyen, gelişen olay ve hareketler dizisi, proses

Yazılım süreci nedir?

- Bir ***yazılım ürününü*** üretmeyi sağlayan birbiriyle ***tutarlı aktivite grubudur.***

Yazılımı baştan geliştirme

Piyasada satılan hazır yazılımları yapılandırma ya da tümleştirme

Yazılım süreci nedir?

- Ne yapılmak istendiğini tüm uygulama detaylarına girmeden tanımlar.
- Yazılım süreci bizim yazılım üretme yolumuzdur. *

- *Kaynak: Object-Oriented and Classical SWE, 7thEdition, Stephen R. Schach, p71.

Süreç Modelleri

- Süreç Modelleri, Yazılım Yaşam Döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ya da sırada, nasıl uygulanacağını tanımlar.
- Yazılım geliştirmenin bahsedilen zorluklarıyla bahsedebilmek için, geliştirmeyi sistematik hale getirmeyi hedefleyen çeşitli süreç modelleri ortaya çıkmıştır.
  - **Bu modellerin temel hedefi;** proje başarısı için, yazılım geliştirme yasam döngüsü ("software development life cycle") boyunca izlenmesi önerilen mühendislik süreçlerini tanımlamaktır.
- Modellerin ortaya çıkmasında, ilgili dönemin donanım ve yazılım teknolojileri ile sektör ihtiyaçları önemli rol oynamıştır.
- Süreçlerin içsel ayrıntıları ya da süreçler arası ilişkilerle ilgilenmez.
- Özetle yazılım üretim işinin genel yapılma düzenine ilişkin rehberler olarak kullanılabilir.
- **Örnek:**
  - Geleneksel modeller (Çaglayan ("waterfall") , evrimsel, döngüsel …vb.)
  - Çevik ("Agile") Modeller (Uçdeger ("extreme") programlama modeli – XP)

Süreç Modelleri Neden Önemlidir?

- Endüstri kaliteye önem vermektedir.
  - (örn. performans, üretkenlik)

- Deneyimler göstermektedir ki süreçlerin ürünlerin kalitesine kayda değer etkisi vardır.
  - Ürünlerin istenen kalitede olmasını süreçleri kontrol ederek daha iyi sağlayabiliriz.

- Yönetici ve geliştiricilerin, ***yazılım geliştirme sürecinin karışıklığı ile baş etmelerini*** sağlarlar .

Süreç Modelleri

Düzenleyici Süreç Modelleri

- Kodla ve Düzelt (Code and Fix)
- Gelişigüzel Model
- Barok Modeli
- Çağlayan/Şelale Modeli (Waterfall Model)
- V Modeli (V-shaped Model)
- Prototipleme
- Helezonik Model (Spiral Model)
- Evrimsel Geliştirme Modeli (Evolutionary Development)
- Artırımsal Geliştirme Modeli (Incremental Development)
- Araştırma Tabanlı Model (Resource Based Model)
- Formal Sistem Geliştirme (Formal System Development)
- Bileşen Tabanlı Geliştirme (Component Based Development)

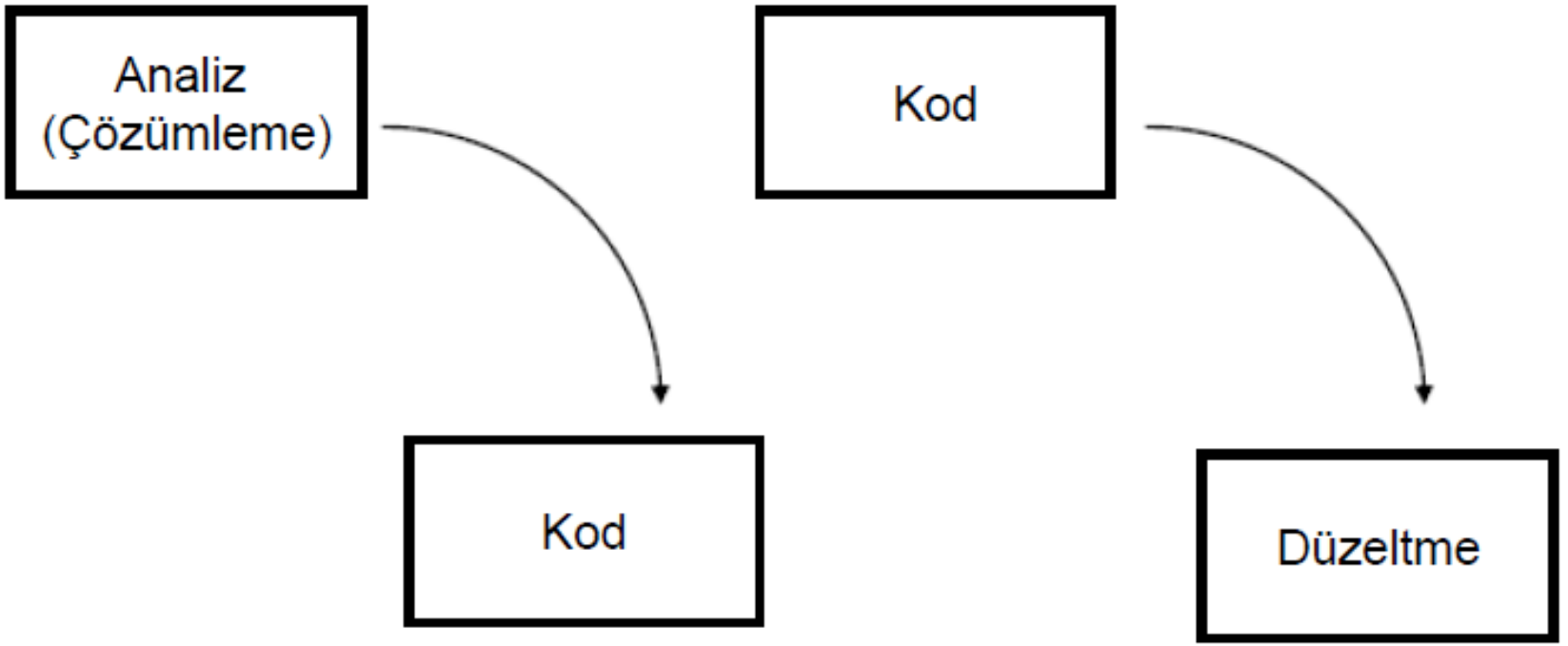
Birleşik Süreç

- Birleşik Sürecin Aşamaları

Çevik Yazılım Süreci

- Uçdeger Programlama (“Extreme Programming – XP”)
- Scrum
- Özellik Güdümlü Gelistirme (“Feature-Driven Development – FDD”)
- Çevik Tümlesik Süreç (“Agile Unified Process – AUP”)

Kodla ve Düzelt (Code and Fix)



Kodla ve Düzelt - Avantajları

- Tüm gereken yeterli olacak kadar gayrettir.
- Tüm adımlardaki gayret direk olarak ürüne katkı sağladığından çoğu müşteri ödeme yapmaktan mutlu olur.
- Eğer ürün onu yapanlar tarafından kullanılacaksa avantajlıdır.

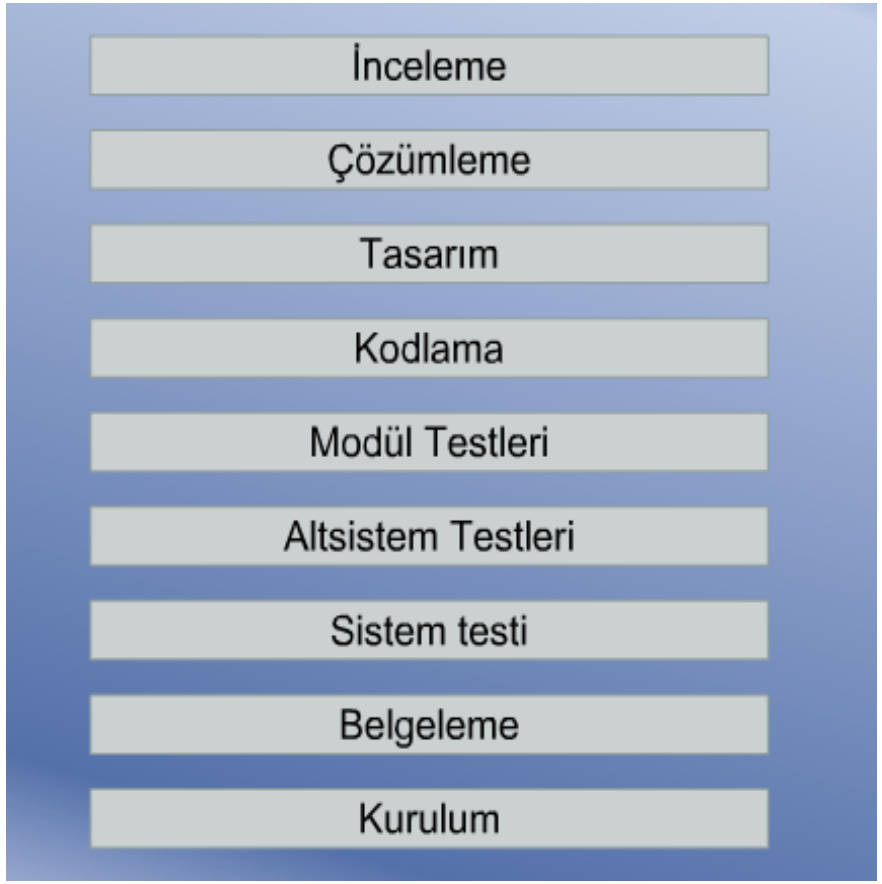
Kodla ve Düzelt - Dezavantajları

- Kodlamaya başlamadan önce değişiklik tahmin edilmediğinden, birbirini izleyen değişikliklerden sonra kod karmakarışık bir hale gelir ve daha sonraki düzeltmeleri yapmak daha da zorlaşır.
- Geliştirilen sistemin boyutunun artması, yapısal olmayan bir şekilde karmaşıklığının yönetilmesini zorlaştırır.
- Müşterinin sürece dahil edilmemesi kullanıcı ihtiyaçlarına uygun olmamasına yol açar.
- Bireysel geliştiriciler için uygundur, takımlar için değil.

Gelişigüzel Model

- Geliştirme ortamında herhangi bir model ya da yöntem kullanılmaz.
- Geliştiren kişiye bağımlı (belli bir süre sonra o kişi bile sistemi anlayamaz ve geliştirme güçlüğü yaşar).
- İzlenebilirliği ve bakımı oldukça zor.
- 60'lı yıllarda, daha çok tek kişilik üretim ortamlarında kullanılan yöntemlerdir.
- Yani basit programlama yöntemidir..

Barok Modeli



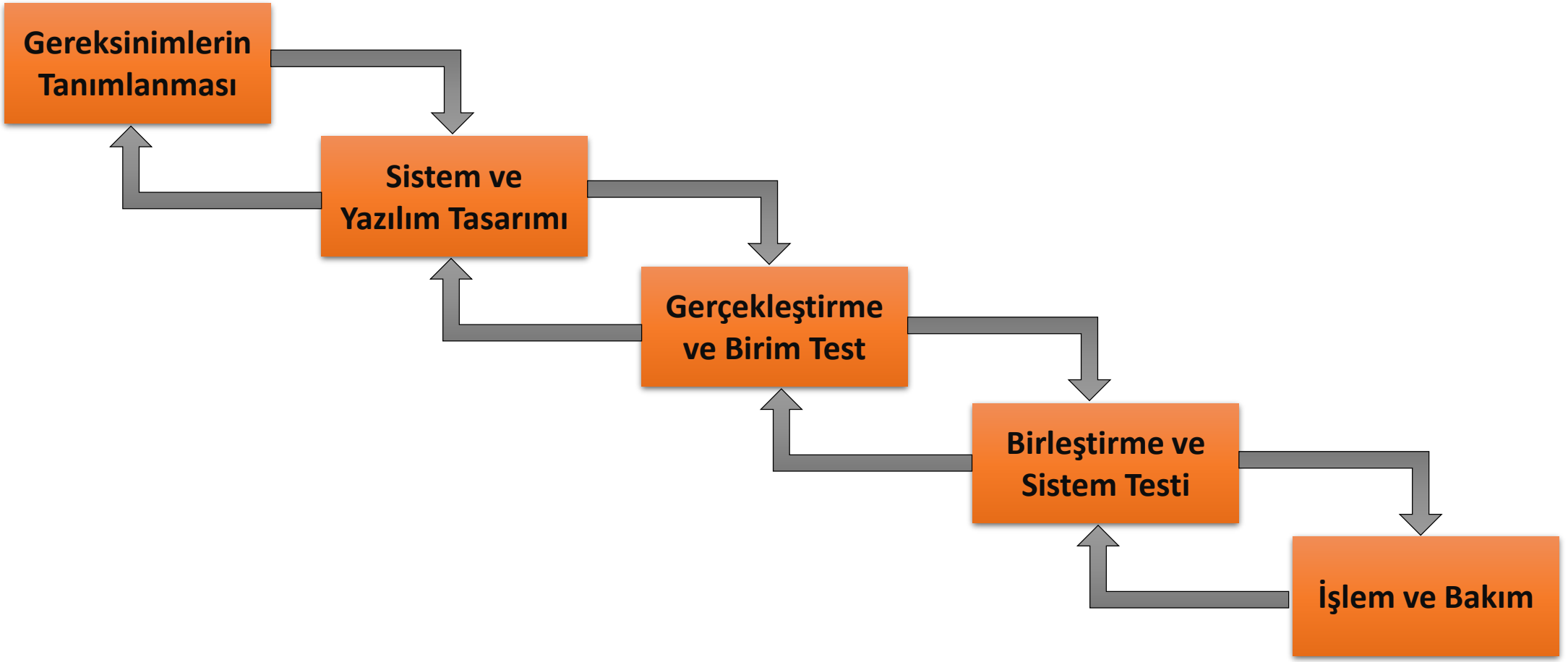
- Yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği model.
- Barok modeli 70'li yılların ortalarından başlanarak kullanılmaya başlanmıştır.
- Belgelemeyi ayrı bir süreç olarak ele alır, ve yazılımın geliştirilmesi ve testinden hemen sonra yapılmasının öngörür.
- Halbuki, günümüzde belgeleme yapılan işin doğal bir ürünü olarak görülmektedir.
- Aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değil.

Gerçekleştirim aşamasına daha fazla ağırlık veren bir model olup, günümüzde kullanımı önerilmemektedir.

Çağlayan Modeli

- Yaşam döngüsü temel adımları baştan sona en az bir kez izleyerek gerçekleştirilir.
- İyi tanımlı projeler ve üretimi az zaman gerektiren yazılım projeleri için uygun bir modeldir.
- Geleneksel model olarak da bilinen bu modelin kullanımı günümüzde giderek azalmaktadır.
- Barok modelin aksine belgeleme işlevini ayrı bir aşama olarak ele almaz ve üretimin doğal bir parçası olarak görür.
- Barok modele göre geri dönüşler iyi tanımlanmıştır.
- Yazılım tanımlamada belirsizlik yok (ya da az) ise ve yazılım üretimi çok zaman almayacak ise uygun bir süreç modelidir.
- Bir sonraki aşama, önceki aşama tamamlanmadan başlayamaz.
- Her aşamanın sonucu bir ya da birden fazla onaylanan (imzalanan) belgedir.
- Gerektiğinde geliştirme aktivitelerinde iterasyonlar (tekrarlamalar) olabilir.

Çağlayan/Şelale Modeli (Waterfall Model)



Çağlayan Modeli - Aşamaları

- **Gereksinim Tanımlama:** Gerçekleştirilecek sistemin gereksinimlerinin belirlenmesi isidir.
  - Müşteri ne istiyor? Ürün ne yapacak, ne işlevsellik gösterecek?
- **Sistem ve Yazılım Tasarımı:** Gereksinimleri belirlenmiş bir sistemin yapısal ve detay tasarımını oluşturma isidir.
  - Ürün, müşterinin beklediği işlevselliği nasıl sağlayacak?
- **Gerçekleştirme ve Birim Test:** Tasarımı yapılmış bir yazılım sisteminin kodlanarak gerçekleştirilmesi isidir.
  - Yazılım ürünü, tasarımı gerçekleştirecek şekilde kodlandı mı?
- **Birleştirme ve Sistem Testi:** Gerçekleştirilmiş sistemin beklenen işlevselliği gösterip göstermediğini sınama işlemidir.
  - Ürün, müşterinin beklediği işlevselliği sağlıyor mu?
- **İşlem ve Bakım:** Müşteriye teslim edilmiş ürünü, değişen ihtiyaçlara ve ek müşteri taleplerine göre güncelleme isidir.
  - Ürün müşteri tarafından memnuniyetle kullanılabiliyor mu?

Çağlayan Modeli - Avantajları

- Müşteriler ve son kullanıcılar tarafından da iyi bilenen anlaşılabilen adımlardan oluşur.
- İterasyonlar (tekrarlamalar) bir sonraki ve bir önceki adımlarla gerçekleşir, daha uzak adımlarla olması nadirdir.
- Değişiklik süreci yönetilebilir birimlere bölünmüştür.
- Gereksinim adımı tamamlandıktan sonra sağlam bir temel oluşur.
- Erken işin miktarını arttırır.

Çağlayan Modeli - Avantajları

- Proje yöneticileri için işin dağılımını yapma açısından kolaydır.
- Aşamaları iyi anlaşılabilir.
- Gereksinimleri iyi anlaşılabilen projelerde iyi çalışır.
- Kalite gereksinimlerinin bütçe ve zaman kısıtlamasında göre çok daha önemli olduğu projelerde iyi çalışır.

Çağlayan Modeli Problemleri

Problem - 1

Test aşaması geliştirme sürecinin en sonunda yapılır. Hatalar önemli yeniden tasarım gerekliliğini oluşturur.

Çözüm

1. Çözümleme aşamasının önüne bir ön-tasarım aşaması eklenir böylece programlama kısıtlamaları önceden anlaşılabilir.
2. Her aşamanın sonunda genişletilmiş belgelendirme yapılır

Neden?

Erken aşamalarda tasarım= belgelendirme
Etkili yeniden tasarıma izin verir
Proje ile ortak anlayış

Çağlayan Modeli Problemleri

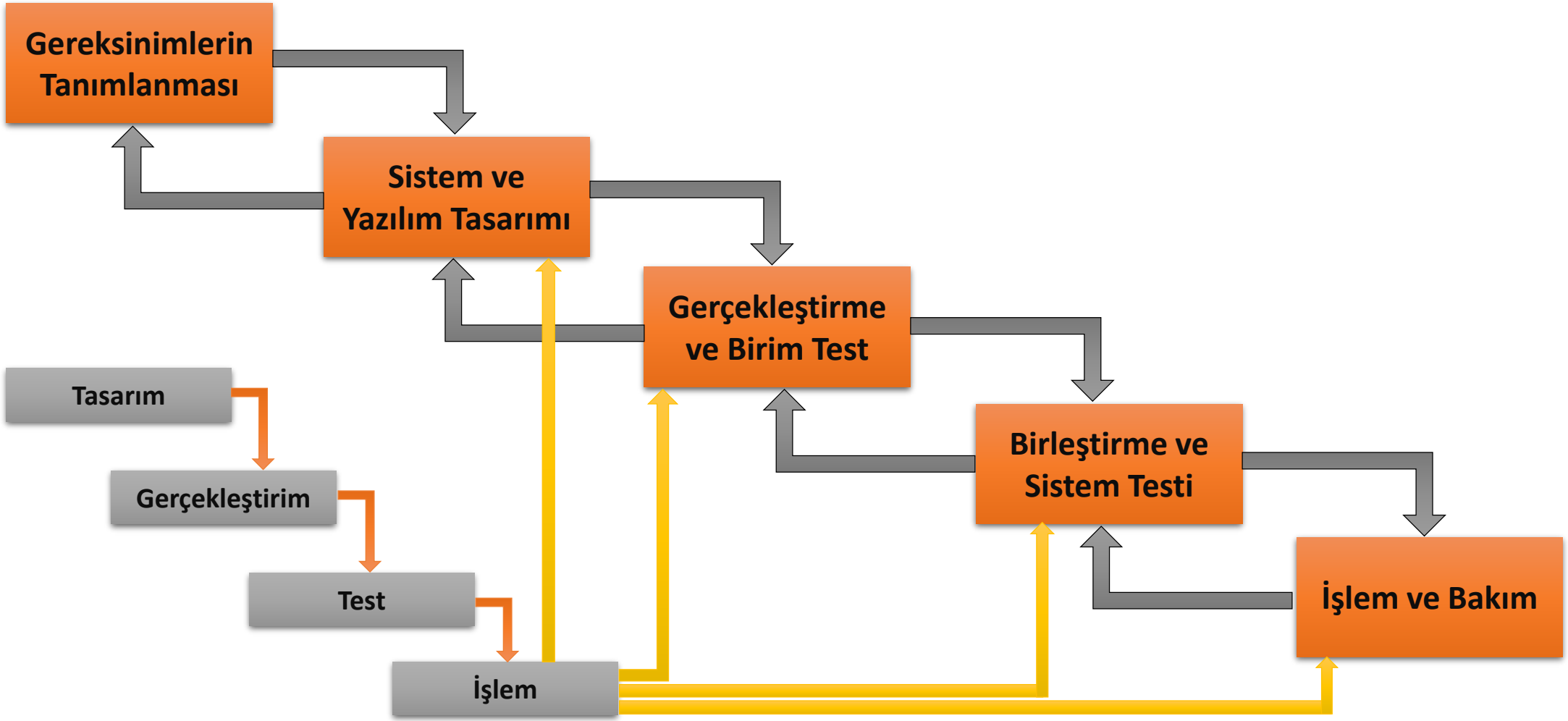
Problem - 2

Eğer ürün tamamıyla orijinal ise, sistemi yapmadan önce biraz deneysel testlerin yapılması gereklidir.

Çözüm

Bazı anahtar hipotezleri sınamak için bir prototip yap.

Prototip yaptıktan sonra



Çağlayan Modeli Problemleri

Problem - 3

Önceden anlaşma sağlansa bile yazılımın ne yapacağı konusu yoruma açıktır.
Kullanıcılar kaliteyi en sondan önce anlayamazlar

Çözüm

Teslim etmeden önce sürece müşteriyi de dahil et.

- Gözden geçirmeler

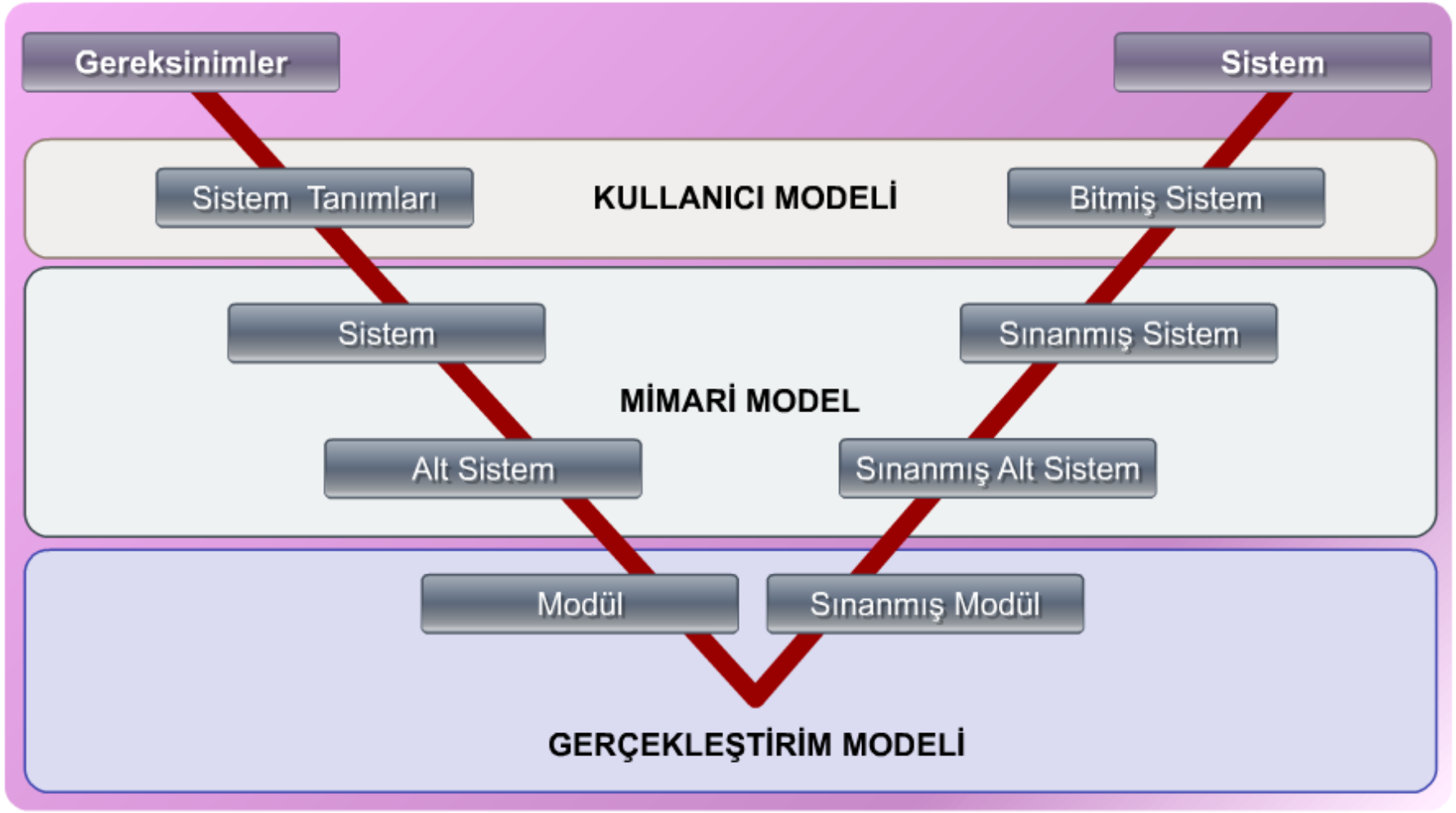
Çağlayan Modeli - Diğer dezavantajları

- Bitirme kriteri olarak belgelendirmeye önem verilmektedir.
  - Bazı alanlar için mümkünken (derleyiciler, işletim sistemleri, vb.) etkileşimli son kullanıcı uygulamaları gibi alanlar için zordur.
- Sistem geliştirilmesi süresince de gereksinimler sıklıkla değişir.
  - Çağlayan modeli gereksinimlerin çok iyi anlaşılabildiği durumlarda kullanılmalıdır.
- İki ya da daha önceki fazlara gitmek çok maliyetlidir.
  - bu durumda da gerektiğinde tüm fazı yeniden gerçekleştirmek çok büyük bir iştir.
- Bir faz tamamlanmadan diğerine geçilememesi riski arttırır.

V Modeli (V-shaped Model)

- Proje ve gereksinim planlaması
  - Ürün gereksinimleri ve belirtim çözümlemesi
    - Mimari ve yüksek seviye tasarım
      - Detaylı tasarım
        - Kodlama
      - Birim testi
    - Tümleştirme ve test
  - Sistem ve kabul edilme testleri
- Üretim, işletim ve sürdürülebilirlik

V Modeli


Gereksinimler
Sistem
Sistem Tanımları
KULLANICI MODELİ
Bitmiş Sistem
Sistem
Sınanmış Sistem
MİMARİ MODEL
Alt Sistem
Sınanmış Alt Sistem
Modül
Sınanmış Modül
GERÇEKLEŞTİRİM MODELİ

V Modeli

- Sol taraf üretim, sağ taraf sınama işlemleridir.
- V süreç modelinin temel çıktıları;

Kullanıcı Modeli

Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.

Mimari Model

Sistem tasarımı ve oluşacak altsistem ile tüm sistemin sınama işlemlerine ilişkin işlevler.

Gerçekleştirim Modeli

Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar.

V Modeli

- Belirsizliklerin az, iş tanımlarının belirgin olduğu BT projeleri için uygun bir modeldir.
- Model, kullanıcının projeye katkısını arttırmaktadır.
- BT projesinin iki aşamalı olarak ihale edilmesi için oldukça uygundur:
  - İlk ihalede kullanıcı modeli hedeflenerek, iş analizi ve kabul sınamalarının tanımları yapılmakta,
  - İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modeli tasarlanıp, gerçeklenmektedir.

V Modeli - Avantajları

- Verification ve validation planları erken aşamalarda vurgulanır.
- Verification ve validation sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır.
- Proje yönetimi tarafında takibi kolaydır
- Kullanımı kolaydır.

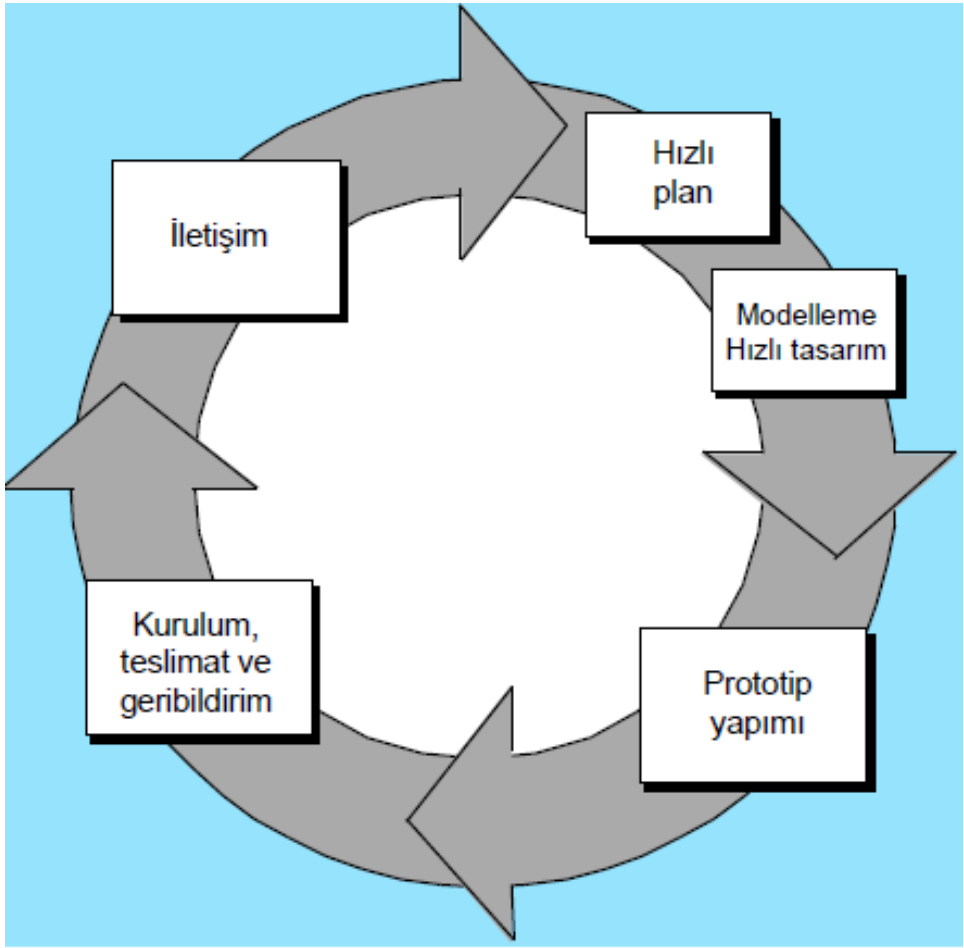
V Modeli - Dezavanatjları

- Aynı zamanda gerçekleştirilebilecek olaylara kolay imkan tanımaz.
- Aşamalar arasında tekrarlamaları kullanmaz.
- Risk çözümleme ile ilgili aktiviteleri içermez

Prototipleme

- Gereksinim tanımlama fazında hızlıca yapılan kısmi gerçekleştirme.
- Gereksinimler netleştikçe prototipi düzelt.
- Müşteri memnun olana kadar düzeltmelere devam et.

Prototipleme



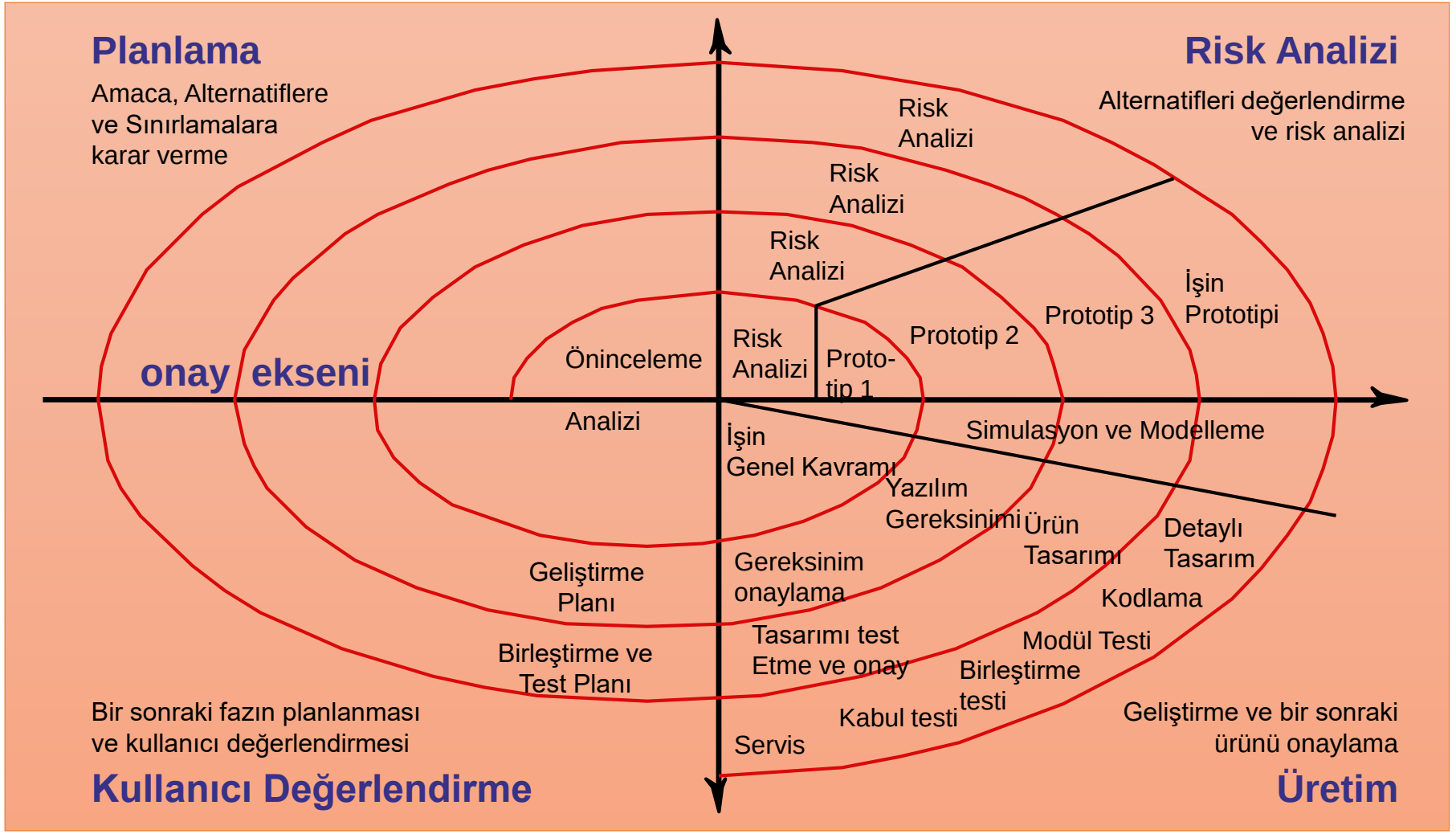
Prototipleme - Avantajları

- Kullanıcı sistem gereksinimlerini görebilir.
- Karmaşa ve yanlış anlaşılmaları engeller.
- Yeni ve beklenmeyen gereksinimler netleştirilebilir.
- Risk kontrolü sağlanır.

Prototipleme - Dezavantajları

- Belgelendirmesi olmayan hızlı ve kirli (quick and dirty) prototipler.
- Prototip hedefleri net değilse kod hackleme ya da jenga başlar.
- Düzeltme aşaması atlanırsa, düşük performansa yol açar.
- Müşteri prototipten de son ürün gibi görünüm ve etki bekler.

Helezonik Model (Spiral Model)



Helezonik Model - Aşamaları

- **Planlama**
  - Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirme
- **Risk Analizi**
  - Risk seçeneklerinin araştırılması ve risklerin belirlenmesi
- **Üretim**
  - Ara ürünün üretilmesi
- **Kullanıcı Değerlendirmesi**
  - Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler

Helezonik Model

- Risk Analizi Olgusu ön plana çıkmıştır.
- Hedefler, alternatifler ve kısıtlamalar belirlenir.
- Alternatifler değerlendirilir, riskler belirlenip çözülür.
- Aşamanın ürünü geliştirilir.
- Sonraki aşama planlanır.
- Her döngü bir aşamayı ifade eder. Doğrudan tanımlama, tasarım,... vs. gibi bir aşama yoktur.
- Yinelemeli artımsal bir yaklaşım vardır.
- Prototip yaklaşımı vardır.

Helezonik Geliştirme

- Süreç arka arkaya devam eden sıralı aktiviteler şeklinde gösterilmek yerine spiral şekilde gösterilir.
- Spiral üzerindeki her bir halka bir fazı gösterir.
- Belirtim, tasarım gibi kesin fazlar yoktur – spiral deki halkalar neye ihtiyaç varsa onu gerçekleştirmek için seçilir.
- Süreç boyunca risklerin değerlendirilmesi ve çözümü açık olarak yapılır.

Helezonik Model - Avantajları

- Kullanıcılar sistemi erken görebilirler.
- Geliştirmeyi küçük parçalara böler . En riskli kısımlar önce gerçekleştirilir.
- Pek çok yazılım modelini içinde bulundurur.
- Riske duyarlı yaklaşımı potansiyel zorlukları engeller.
- Seçeneklere erken dikkate odaklanır.
- Hataları erken gidermeye odaklanır.
- Yazılım-donanım sistemi geliştirme için bir çerçeve sağlar.

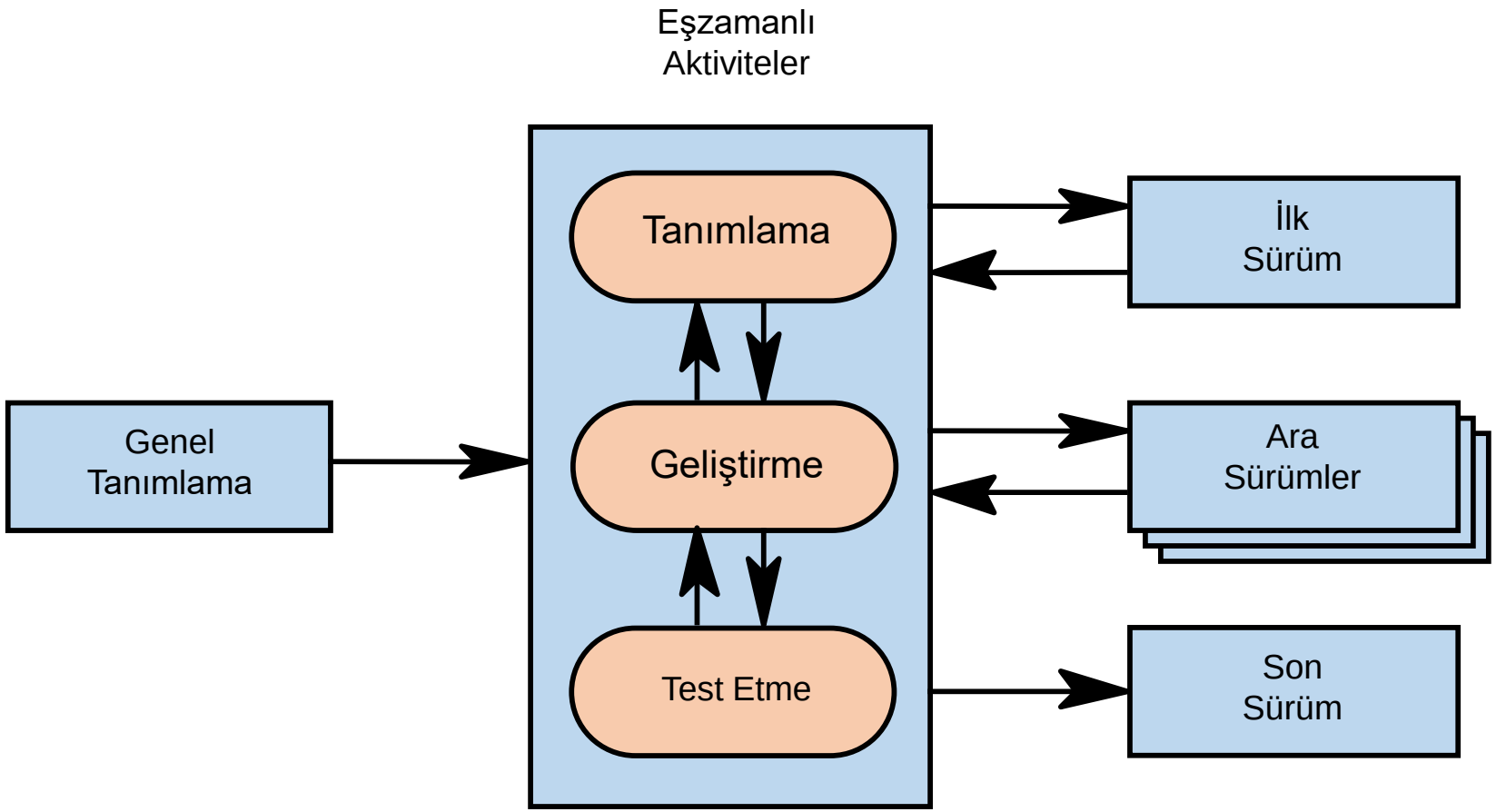
Helezonik Model - Problemler

- Küçük ve düşük riskli projeler için pahalı bir yöntemdir.
- Komplekstir (karmaşık).
- Spiral sonsuza gidebilir.
- Ara adımların fazlalığı nedeniyle çok fazla dokümantasyon gerektirir.
- Büyük ölçekte projeler
- Kontrat tabanlı yazılıma uymaz.
  - Yazılımın içten geliştirileceğini varsayar.
  - Kontrat tabanlı yazılımlar adım adım anlaşma esnekliğini sağlamaz.
- Öznel risk değerlendirme deneyimine dayanır.
  - Yüksek riskli öğelere yoğunlaşmak, yüksek riskli öğelerin doğru belirlenmesini gerektirir.

Evrimsel Geliştirme Modeli (Evolutionary Development Model)

- İlk tam ölçekli modeldir.
- Anahtar gereksinimleri ile başlangıç sistemi geliştirilir.
- Müşteri geribildirimi ile sitem pek çok versiyonla yavaş yavaş geliştirilir.
- Belirtim (specification), geliştirme ve geçerleme (validation) aktivitleri koşut zamanlı yürütülür.
- Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir (banka uygulamaları).
- Her aşamada üretilen ürünler, üretildikleri alan için tam işlevselliği içermektedirler.
- Pilot uygulama kullan, test et, güncelle diğer birimlere taşı.
- Modelin başarısı ilk evrimin başarısına bağımlıdır.

Evrimsel Geliştirme Modeli



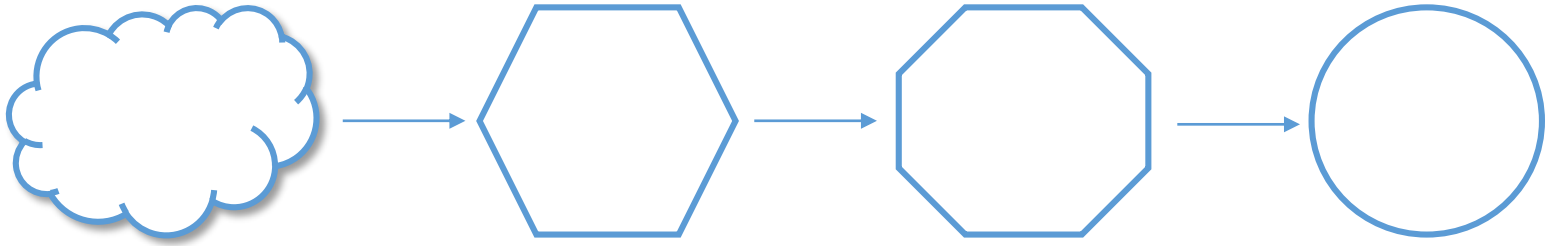
Evrimsel Geliştirme Modeli

- **İki çeşit evrimsel geliştirme vardır:**
  - Keşifçi geliştirme (exploratory development)
    - Hedef: Müşterinin gereksinimlerini incelemek için müşteri ile çalışıp son sistemi teslim etmek
    - İyi anlaşılan gereksinimlerle başlanmalıdır.

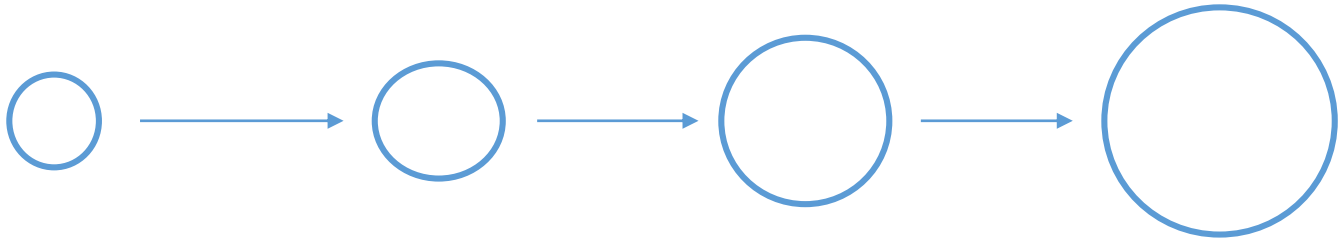
    “ Ne istediğimi sana söyleyemem ama onu gördüğümde bilirim”

  - Atılacak prototipleme (throw-away prototyping)
    - Hedef: Sistem gereksinimlerini anlamak
    - Tam anlaşılmamış gereksinimlerle başlar

Karşılaştırma



Çağlayan Modeli



Evrimsel Geliştirme

Zaman

Evrimsel Geliştirme Modeli - Avantajlar

- Kullanıcıların kendi gereksinimlerini daha iyi anlamalarını sağlar.
- Sürekli değerlendirme erken aşamalardaki geliştirme risklerini azaltır.
- Hatalar azalır.

Evrimsel Geliştirme Modeli - Problemler

- Sürecin görünürlüğü azdır (düzenli teslim edilebilir ürün yoktur).
- Sistemler sıklıkla iyi yapılandırılmaz (sürekli değişiklik yazılımın yapısına zarar verir).
- Bakımı zordur.
- Yazılım gereksinimini yenilemek gerekebilir.

Evrimsel Geliştirme - Uygulanabilirliği

- Küçük ve orta boyutlu etkileşimli sistemler (500.000 LOC dan daha az olan).
- Büyük bir sistemin parçaları (ör. Kullanıcı ara yüzleri).
- Kısa süreli kullanılacak sistemler.

Örnek

- Çok birimli banka uygulamaları.
- Önce sistem geliştirilir ve Şube-1'e yüklenir.
- Daha sonra aksaklıklar giderilerek geliştirilen sistem Şube-2'ye yüklenir.
- Daha sonra geliştirilen sistem Şube-3'e,…. yüklenir.
- Belirli aralıklarla eski şubelerdeki güncellemeler yapılır.

Yazılım Süreç Modellerinde Süreç Tekrarı ("Process Iteration")

- Yazılım süreç modelleri tek bir defada uygulanmak yerine, birkaç tekrarda uygulanabilir.
  - Örneğin, geniş kapsamlı 5 alt sistemden oluşan bir sistemin; ilk alt sistemi için çağlayan modeli uygulandıktan sonra, geri kalanı için çağlayan modeli tekrar uygulanabilir.
  - Bu şekilde geliştirme riskleri en aza indirilerek ilk tekrarda kazanılan deneyimden, sistemin geri kalanı geliştirilirken faydalanılabilir.

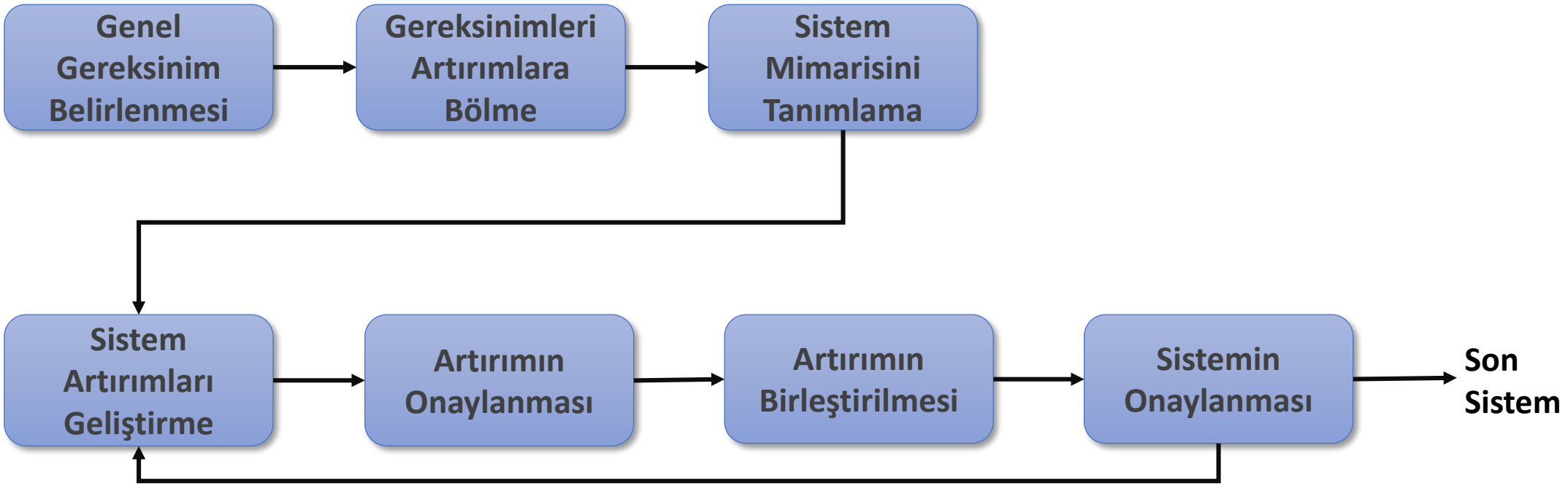
- Hangi süreç modelinin, sistemin hangi bölümleri için ve kaç tekrarda uygulanacağına proje basında karar verilir.

- Süreç tekrarıyla yakından ilişkili iki geleneksel model vardır:
  - Artırımsal ("incremental") model
  - Döngüsel ("spiral") model

Artırımsal Geliştirme Modeli (Incremental Development Model)

- Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir.
- Öğrencilerin bir dönem boyunca geliştirmeleri gereken bir programlama ödevinin 2 haftada bir gelişiminin izlenmesi (bitirme tezleri).
- Uzun zaman alabilecek ve sistemin eksik işlevlikle çalışabileceği türdeki projeler bu modele uygun olabilir.
- Bir taraftan kullanım, diğer taraftan üretim yapılır.

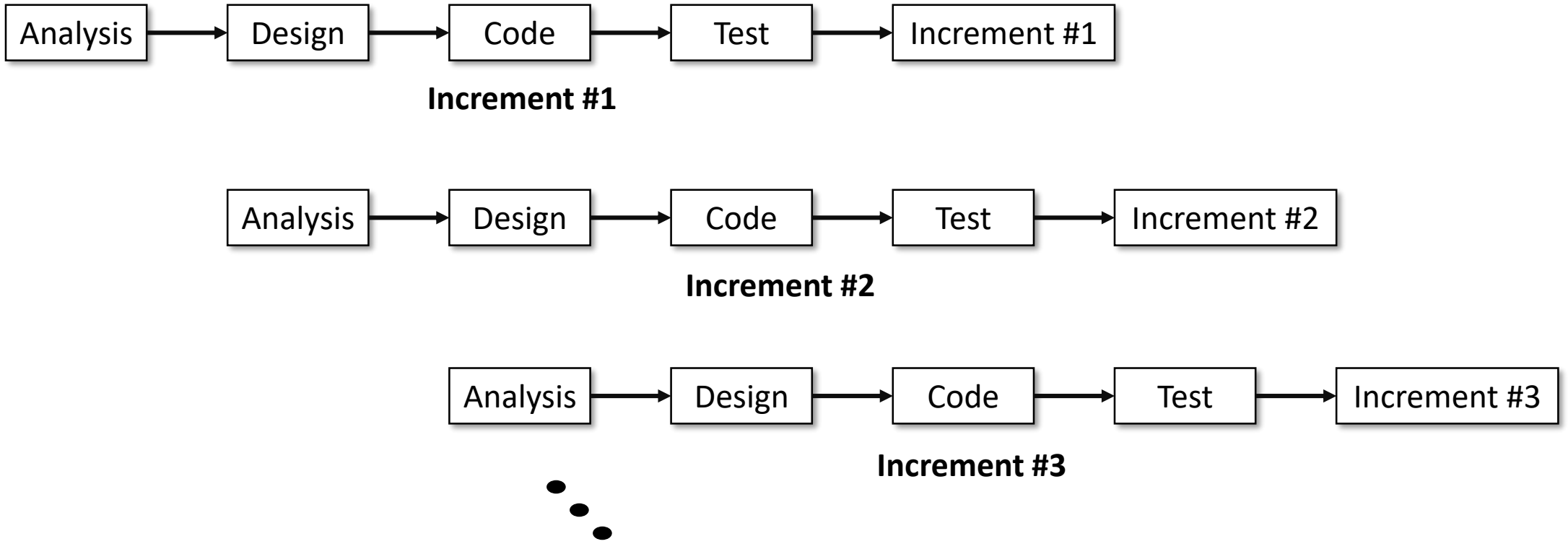
Artırımsal Geliştirme Modeli



Tamamlanmamış Sistem

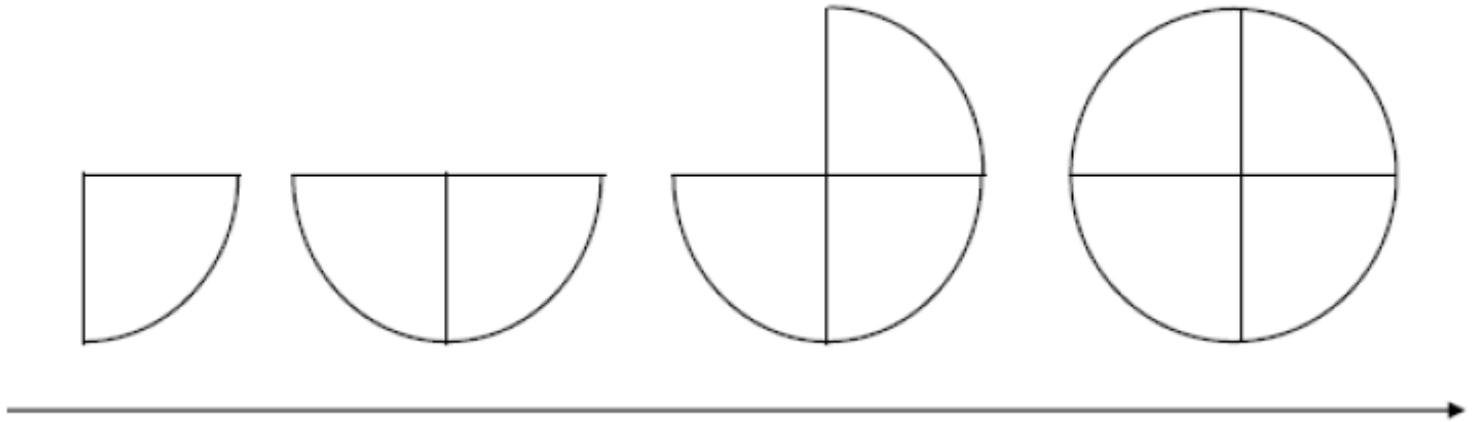
Arttırımsal Geliştirme Modeli

- Aslında Çağlayan modelinin örtüşen şekilde uygulanmasıdır.



Artırımsal Geliştirme Modeli

- Çağlayan modeli ve Evrimsel geliştirme arası bir model:



Artırımsal Geliştirme Modeli - Avantajları

- Sistem için gerekli olan gereksinimler müşterilerle belirlenir
- Gereksinimlerin önemine göre teslim edilecek artımlar belirlenir
- Öncelikle en önemli gereksinimleri karşılayan çekirdek bir sitem geliştirilir.
- Erken artımlar prototip gibi davranarak, gereksinimlerin daha iyi anlaşılmasını sağlar
- Tüm projenin başarısız olma riskini azaltır
- En önemli sistem özellikleri daha fazla sınanma (test edilme) imkanı bulmuş olur.
- Divide and Conquer (Böl ve Yönet) yaklaşımıdır

Artırımsal Geliştirme Modeli - Dezavantajları

- Artımları tanımlamak için tüm sistemin tanımlanmasına ihtiyaç vardır.
- Gereksinimleri doğru boyuttaki artımlara atamak bazen zor olabilir.
- Deneyimli personel gerektirir.
- Artımların kendi içlerinde tekrarlamalara izin vermez.

Araştırma Tabanlı Model

- Yap-at prototipi olarak ta bilinir.
- Araştırma ortamları bütünüyle belirsizlik üzerine çalışan ortamlardır.
- Yapılan işlerden edinilecek sonuçlar belirgin değildir.
- Geliştirilen yazılımlar genellikle sınırlı sayıda kullanılır ve kullanım bittikten sonra işe yaramaz hale gelir ve atılır.
- Model-zaman-fiyat kestirimi olmadığı için sabit fiyat sözleşmelerinde uygun değildir.

Örnek

- En Hızlı Çalışan asal sayı test programı!
- En Büyük asal sayıyı bulma programı!
- Satranç programı!

Formal Sistem Geliştirme
(Formal System Development)

- Cleanroom yazılım geliştirme
- Matematiksel belirtimin farklı gösterim şekilleri ile çalıştırılabilir programa dönüştürülmesine dayalıdır.
- Formal belirtim, tasarım ve geçerleme kullanarak yazılımda doğruluğun geliştirilmesini vurgular.
- Yazılım artımlarla geliştirilir.
- Sürekli tümleştirme vardır ve fonksiyonellik tümleştirilen yazılım artımları ile artar.
- Felsefesi pahalı hata ayıklama işlemini engellemek için kodu ilk yazarken doğru yazmak ve test aşamasından doğruluğunu sağlamak
    - Formal yöntemler
    - Z dili

Formal Sistem Geliştirme

- **Problemleri**
  - Teknikleri uygulayabilmek için eğitim ve özel beceriler gerekmektedir
  - Kullanıcı arayüzü gibi sistemin bazı kısımlarını formal olarak belirtmek zordur.

- **Uygulanabilirliği**
  - Sistem kullanıma konmadan emniyet ve güvenlik durumlarını sağlanması gereken kritik sistemler.

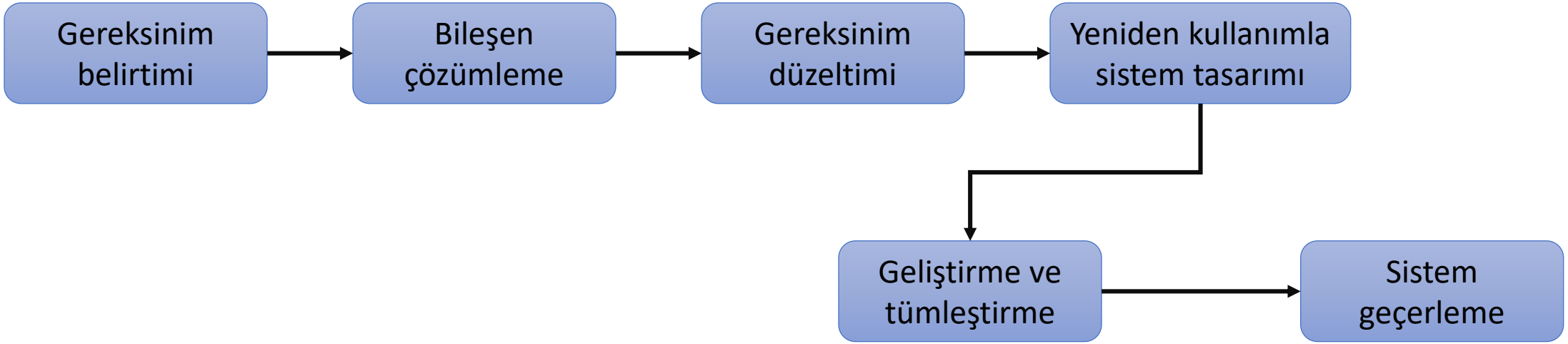
Bileşen-Tabanlı Model (Component-Based Model)

- Sistemin COTS ("commercial-off-the-shelf") adı verilen hazır bileşenler kullanılarak tümleştirilmesi esasına dayanır.

- **Süreç adımları:**
  - Bilesen analizi
  - Gereksinim günleme
  - Bileşenler kullanarak sistem tasarımı
  - Geliştirme ve tümleştirme

- Bu yaklaşım, bilesen standartlarındaki gelişmeler ilerledikçe daha yaygın olarak kullanılmaya başlanmıştır. ama halen kullanımı limitlidir.

Bileşen-Tabanlı Model - Adımlar



Metodolojiler

- **Metodoloji:** Bir BT projesi ya da yazılım yaşam döngüsü aşamaları boyunca kullanılacak ve birbirleriyle uyumlu yöntemler bütünü.

- Bir metodoloji,
  - bir süreç modelini ve
  - belirli sayıda belirtim yöntemini içerir

- Günümüzdeki metodolojiler genelde Çağlayan ya da Helezonik modeli temel almaktadır

Bir Metodolojide Bulunması Gereken Temel Bileşenler (Özellikler)

- Ayrıntılandırılmış bir süreç modeli
- Ayrıntılı süreç tanımları
- İyi tanımlı üretim yöntemleri
- Süreçlerarası arayüz tanımları
- Ayrıntılı girdi tanımları
- Ayrıntılı çıktı tanımları
- Proje yönetim modeli
- Konfigürasyon yönetim modeli
- Maliyet yönetim modeli
- Kalite yönetim modeli
- Risk yönetim modeli
- Değişiklik yönetim modeli
- Kullanıcı arayüz ve ilişki modeli
- Standartlar

Bir Metodoloji Örneği

- Yourdan Yapısal Sistem Tasarımı Metodolojisi.
- Kolay uygulanabilir bir model olup, günümüzde oldukça yaygın olarak kullanılmaktadır.
- Çağlayan modelini temel almaktadır.
- Bir çok CASE aracı tarafından doğrudan desteklenmektedir.

Yourdon Yapısal Sistem Tasarım Metodolojisi

Aşama	Kullanılan Yöntem ve Araçlar	Ne için Kullanıldığı	Çıktı
Planlama	Veri Akış Şemaları, Süreç Belirtimleri, Görüşme, Maliyet Kestirim Yöntemi, Proje Yönetim Araçları	Süreç İnceleme Kaynak Kestirimi Proje Yönetimi	Proje Planı
Analiz	Veri Akış Şemaları, Süreç Belirtimleri, Görüşme, Nesne ilişki şemaları Veri	Süreç Analizi Veri Analizi	Sistem Analiz Raporu
Analizden Tasarıma Geçiş	Akışa Dayalı Analiz, Süreç belirtimlerinin program tasarım diline dönüştürülmesi, Nesne ilişki şemalarının veri tablosuna dönüştürülmesi	Başlangıç Tasarımı Ayrıntılı Tasarım Başlangıç Veri tasarımı	Başlangıç Tasarım Raporu
Tasarım	Yapısal Şemalar, Program Tasarım Dili, Veri Tabanı Tabloları	Genel Tasarım Ayrıntılı Tasarım Veri Tasarımı	Sistem Tasarım Raporu

Özet

- Yazılım yaşam döngüsü, herhangi bir yazılımın, üretim aşaması ve kullanım aşaması birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.
- Yazılım Yaşam Döngüsü temel adımları
  - Planlama
  - Çözümleme
  - Tasarım
  - Gerçekleştirim
  - Bakım
- Belirtim Yöntemleri (Software Specification Methods) - bir çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemler.
- Süreç Modelleri (Software Process Models) - yazılım yaşam döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ya da sırada, nasıl uygulanacağını tanımlayan modeller kullanılır.
- Süreç Modelleri, Yazılım Yaşam Döngüsünde belirtilen süreçlerin geliştirme aşamasında, hangi düzen ya da sırada, nasıl uygulanacağını tanımlar.
- Barok Modeli: Belgelemeyi ayrı bir süreç olarak ele alır, ve yazılımın geliştirilmesi ve testinden hemen sonra yapılmasının öngörür.

Özet

- Çağlayan Modeli Aşamaları:
  - Gereksinim Tanımlama
  - Sistem ve Yazılım Tasarımı
  - Gerçekleştirme ve Birim Test
  - Birleştirme ve Sistem Testi
  - İşlem ve Bakım
- V süreç modelinin temel çıktıları;
  - Kullanıcı Model
  - Mimari Model
  - Gerçekleştirim Modeli
- Helezonik Model Aşamaları:
  - Planlama
  - Risk Analizi
  - Üretim
  - Kullanıcı Değerlendirmesi
- İki çeşit evrimsel geliştirme vardır:
  - Keşifçi geliştirme (exploratory development)
  - Atılacak prototipleme (throw-away prototyping)

Özet

- Artırımsal Geliştirme Modeli-Öğrencilerin bir dönem boyunca geliştirmeleri gereken bir programlama ödevinin 2 haftada bir gelişiminin izlenmesi (bitirme tezleri).
- Formal Sistem Geliştirme: Matematiksel belirtimin farklı gösterim şekilleri ile çalıştırılabilir programa dönüştürülmesine dayalıdır.
- Bileşen-Tabanlı Model : Sistemin COTS ("commercial-off-the-shelf") adı verilen hazır bileşenler kullanılarak tümleştirilmesi esasına dayanır.
- Bileşen-Tabanlı Model – Adımlar
    1. Gereksinim belirtimi
    2. Bileşen çözümleme
    3. Gereksinim düzeltimi
    4. Yeniden kullanımla sistem tasarımı
    5. Geliştirme ve tümleştirme
    6. Sistem geçerleme

Sorular

1. Yazılım yaşam döngüsünün temel adımlarını açıklayınız.
2. Süreç modelleri ve belirtim yöntemlerinin önemini belirtiniz.
3. Süreç modeleri ile belirtim yöntemleri arasındaki farklılıkları belirtiniz.
4. Barok modeli tanımlayınız, yararlarını ve aksak yönlerini belirtiniz.
5. Çağlayan modeli tanımlayınız, yararlarını ve aksak yönlerini belirtiniz.
6. Helezonik modeli tanımlayınız, ayırıcı özelliklerini belirtiniz. Yararlarını ve aksak yönlerini açıklayınız.
7. V model kullanılarak geliştirlecek örnek bir proje tanımı yapınız.
8. VP süreç modeli ile geliştirilecek bir projede uygulanabilecek üç prototipleme örneği veriniz.
9. Evrimsel Geliştirme süreç modelinde Konfigürasyon yönetimi ve değişiklik denetimi neden sorundur?
10. Artımsal geliştirme süreç modelini tanımlayınız, yararlı ve aksak yönlerini belirtiniz.
11. Artımsal geliştirme süreç modeli kullanılarak geliştirilecek bir proje örneği veriniz. Gerekçenizi açıklayınız.
12. Araştırma tabanlı süreç modeli için uygun proje örnekleri veriniz.
13. Metodolojiyi tanımlayınız. Bildiğiniz metodoloji örneklerini listeleyiniz.
14. Yourdon Yapısal Sistem Geliştirme Metodolojisini tamamlayınız.
15. Süreç modelleri, belirtim yöntemleri ile metodolojiler arasındaki ilişkiyi belirtiniz.

Ödevler

- Çevik Yazılım Hakkında Araştırma Yapınız.
- Çevik Yazılım Geliştirmenin projelerdeki başarısı hakkında edindiğiniz bilgileri rapor haline getirin.

Kaynaklar

- "Software Engineering A Practitioner's Approach" (7th. Ed.), Roger S. Pressman, 2013.
- "Software Engineering" (8th. Ed.), Ian Sommerville, 2007.
- *"Guide to the Software Engineering Body of Knowledge", 2004.*
- " Yazılım Mühendisliğine Giriş", TBİL-211, Dr. Ali Arifoğlu.
- "Yazılım Mühendisliği" (2. Basım), Dr. M. Erhan Sarıdoğan, 2008, İstanbul: Papatya Yayıncılık.
- Kalıpsiz, O., Buharalı, A., Biricik, G. (2005). Bilgisayar Bilimlerinde Sistem Analizi ve Tasarımı Nesneye Yönelik Modelleme. İstanbul: Papatya Yayıncılık.
- Buzluca, F. (2010) Yazılım Modelleme ve Tasarımı ders notları (http://www.buzluca.info/dersler.html)
- Hacettepe Üniversitesi BBS-651, A. Tarhan, 2010.
- Yazılım Proje Yönetimi, Yrd. Doç. Dr. Hacer KARACAN

- http://blog.alisuleymantopuz.com/2014/08/30/yazilim-mimarisi-ve-tasarimi-nedir/
- http://www.akifsahman.com/?p=175
- https://ece.uwaterloo.ca/~se464/08ST/index.php?src=lecture
- http://info.psu.edu.sa/psu/cis/azarrad/se505.htm
- http://www.metinakbulut.com/YAZILIM-MIMARISI/
- http://ceng.gazi.edu.tr/~hkaracan/source/YPY_H3.pdf
- http://iiscs.wssu.edu/drupal/node/3399
- http://www.cs.toronto.edu/~sme/CSC340F/slides/21-architecture.pdf
- http://www.users.abo.fi/lpetre/SA10/
- http://sulc3.com/model.html
- http://salyangoz.com.tr/blog/2013/11/23/digerleri/yazilim-gelistirme-surec-modelleri-3/