2. Hafta

YMÜ225-YMÜ321
Yazılım Gereksinim Analizi

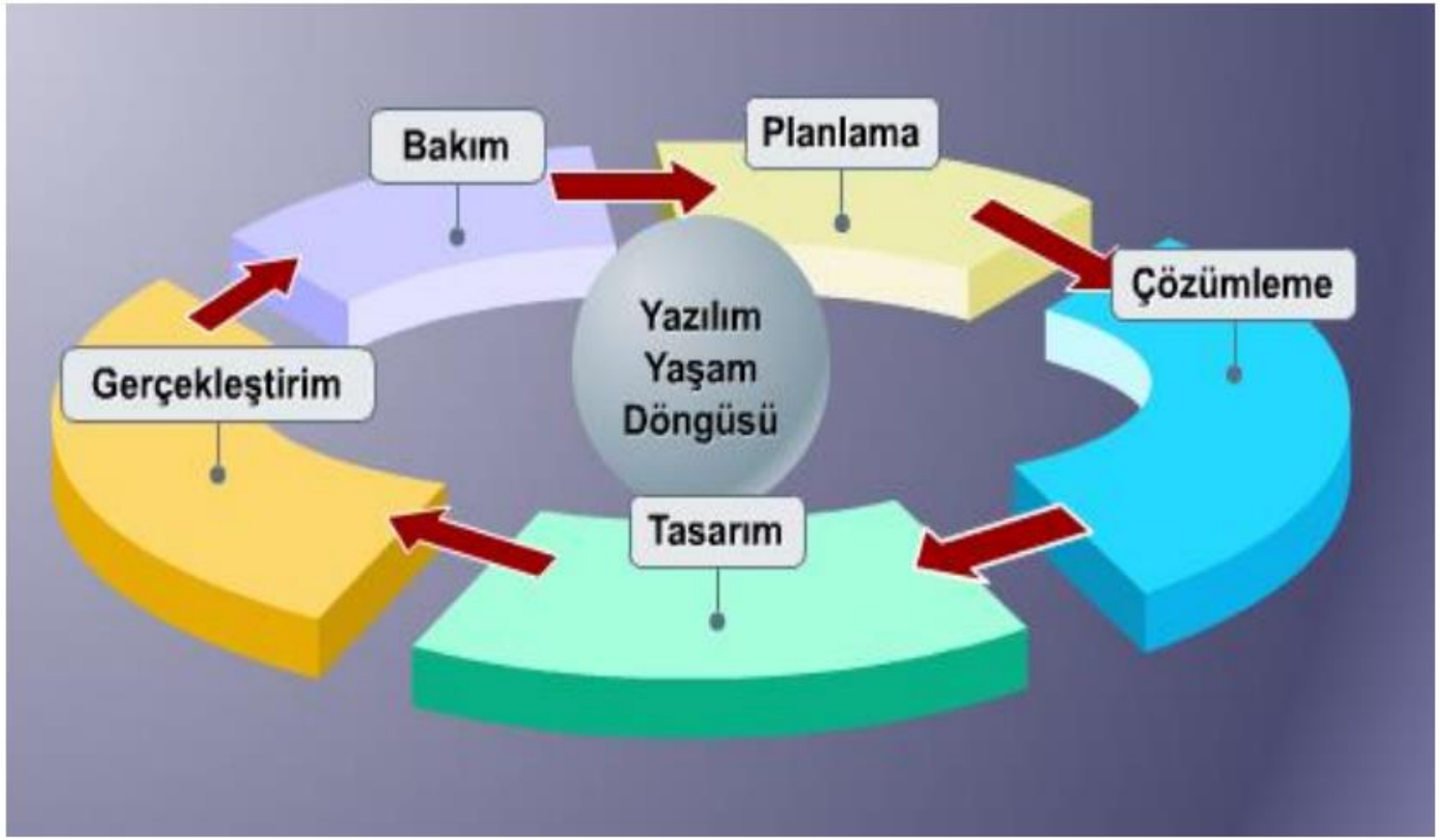
Dr. Feyza Altunbey Özbay

İçerik

- Yazılım Yaşam Döngüsü
- Yazılım Yaşam Döngüsü Temel Adımları
- Belirtim Yöntemleri
- Yazılım Süreci Modelleri

Yazılım Yaşam Döngüsü

- Planlama, çözümleme, tasarım, gerçekleştirim, bakım

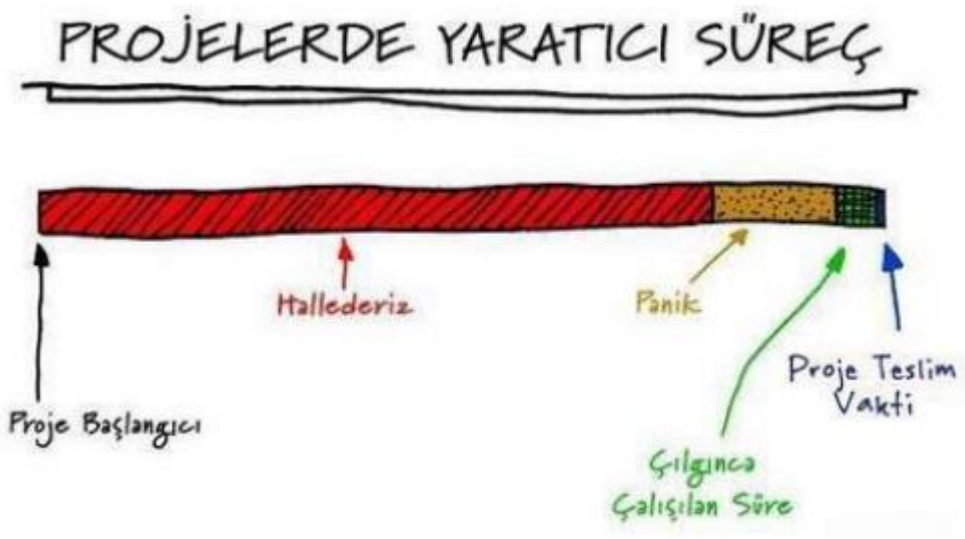


Yazılım Yaşam Döngüsü

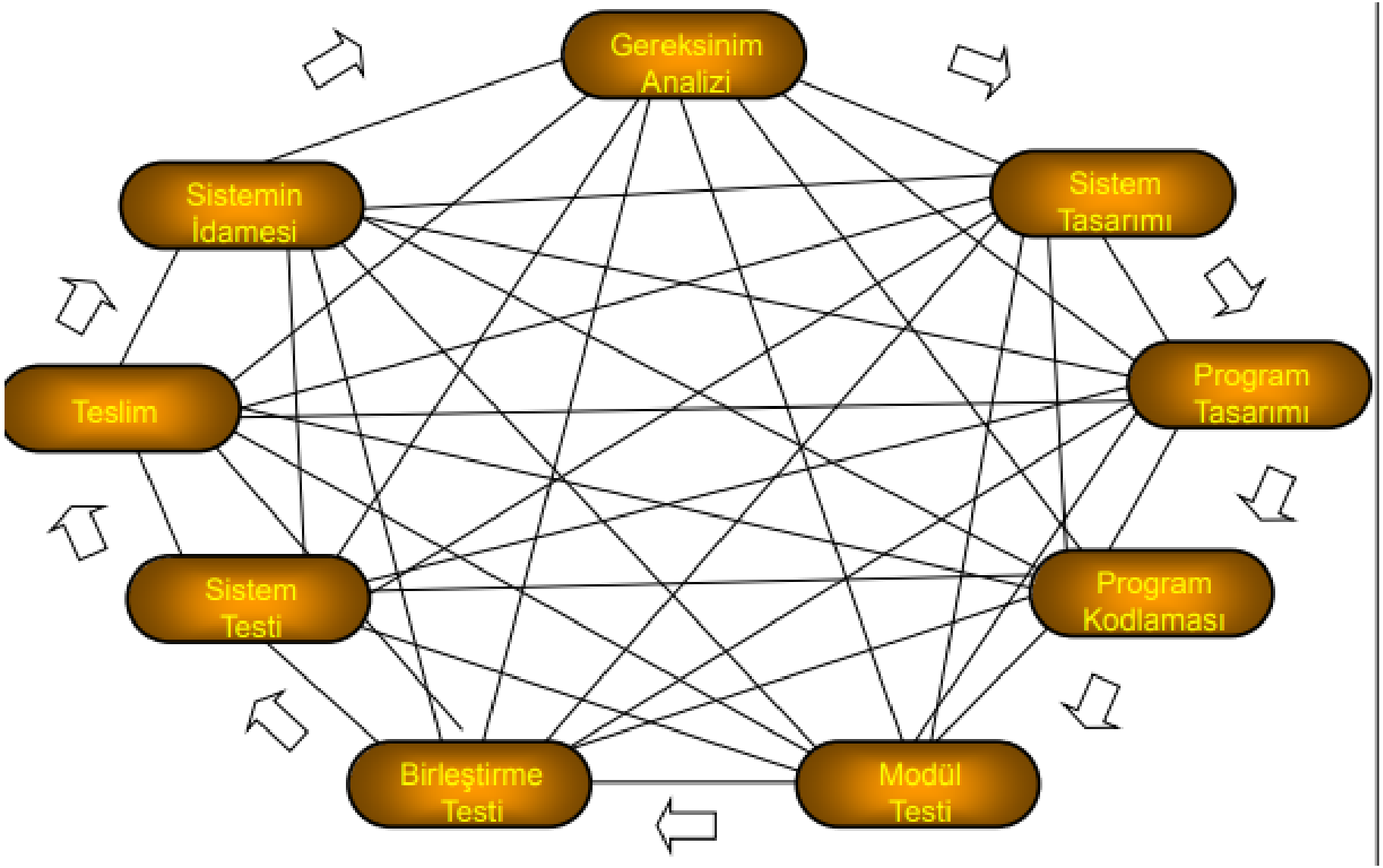
- Herhangi bir yazılımın, **üretim aşaması** ve **kullanım aşaması** birlikte olmak üzere geçirdiği tüm aşamalar biçiminde tanımlanır.
- Yazılım işlevleri ile ilgili gereksinimler sürekli olarak değiştiği ve genişlediği için, söz konusu aşamalar bir döngü biçiminde ele alınır.
- Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur.
- Bu nedenle yazılım yaşam döngüsünün tek yönlü ve doğrusal olduğu düşünülmemelidir.

Yazılım Yaşam Döngüsü-Ne sağlar?

- Mühendislik faaliyetlerinin (üretim, işletme ve bakım aşamaları) yazılıma uyarlanması
- Üretimden kullanıma kadar tüm sürecin fazlara ayrılması ve böylece yönetim kolaylığı
- Her bir aşamada ne yapılacak?
- Her bir aşamadaki standartlara uyumluluk ile kaliteli yazılım geliştirme vb.



Gerçek Hayatta Program Geliştirme



Yazılım Yaşam Döngüsü Temel Adımları

- **Planlama:** Personel ve donanım gereksinimlerinin çıkarıldığı, fizibilite (olurluk-yapılabilirlik) çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.
- **Analiz:** Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak çıkarıldığı aşama. Mevcut var olan işler incelenir, temel sorunlar ortaya çıkarılır.
- **Tasarım:** Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır. mantıksal; önerilen sistemin yapısı anlatılır, olası örgütsel değişiklikler önerilir. fiziksel; yazılımı içeren bileşenler ve bunların ayrıntıları.
- **Gerçekleştirim:** Kodlama, test etme ve kurulum çalışmalarının yapıldığı aşamadır.
- **Bakım:** Hata giderme ve yeni eklentiler yapma aşaması (teslimden sonra).

Yazılım Yaşam Döngüsü Temel Adımları

- Yaşam döngüsünün temel adımları **çekirdek süreçler** (core processes) olarak da adlandırılır.

Bu süreçlerin gerçekleştirilmesi amacıyla

- Belirtim (specification) yöntemleri - bir çekirdek sürece ilişkin fonksiyonları yerine getirmek amacıyla kullanılan yöntemler
- Süreç (process) modelleri - 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).

Yazılım Süreci Modelleri

- Yazılım üretim işinin genel yapılma düzenine ilişkin rehberlerdir.

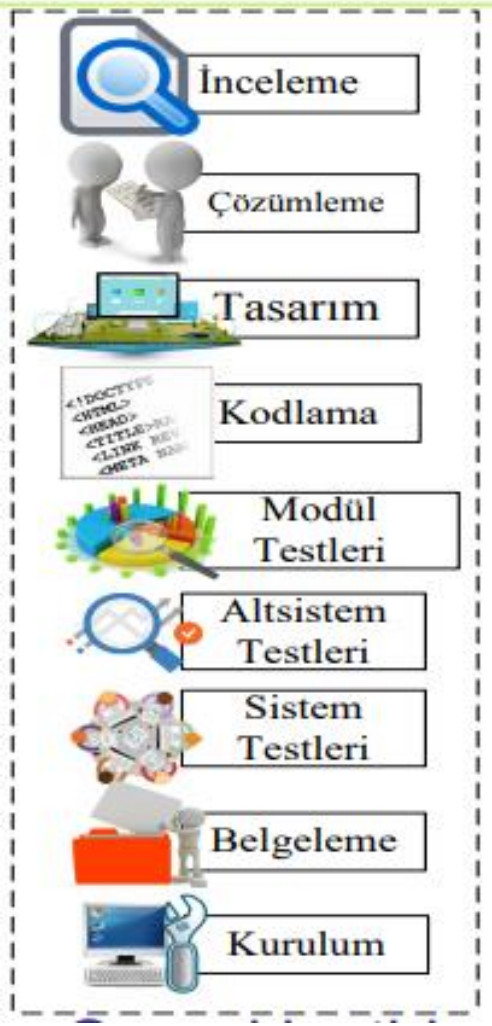
Süreçlere ilişkin ayrıntılarla ya da süreçler arası ilişkilerle ilgilenmezler.

1. Gelişigüzel Model
2. Barok Modeli
3. Çağlayan (Şelale) Modeli
4. V Modeli
5. Helezonik (Spiral) Model
6. Evrimsel Model
7. Artırımsal Model
8. Araştırma Tabanlı Model

Gelişigüzel Model

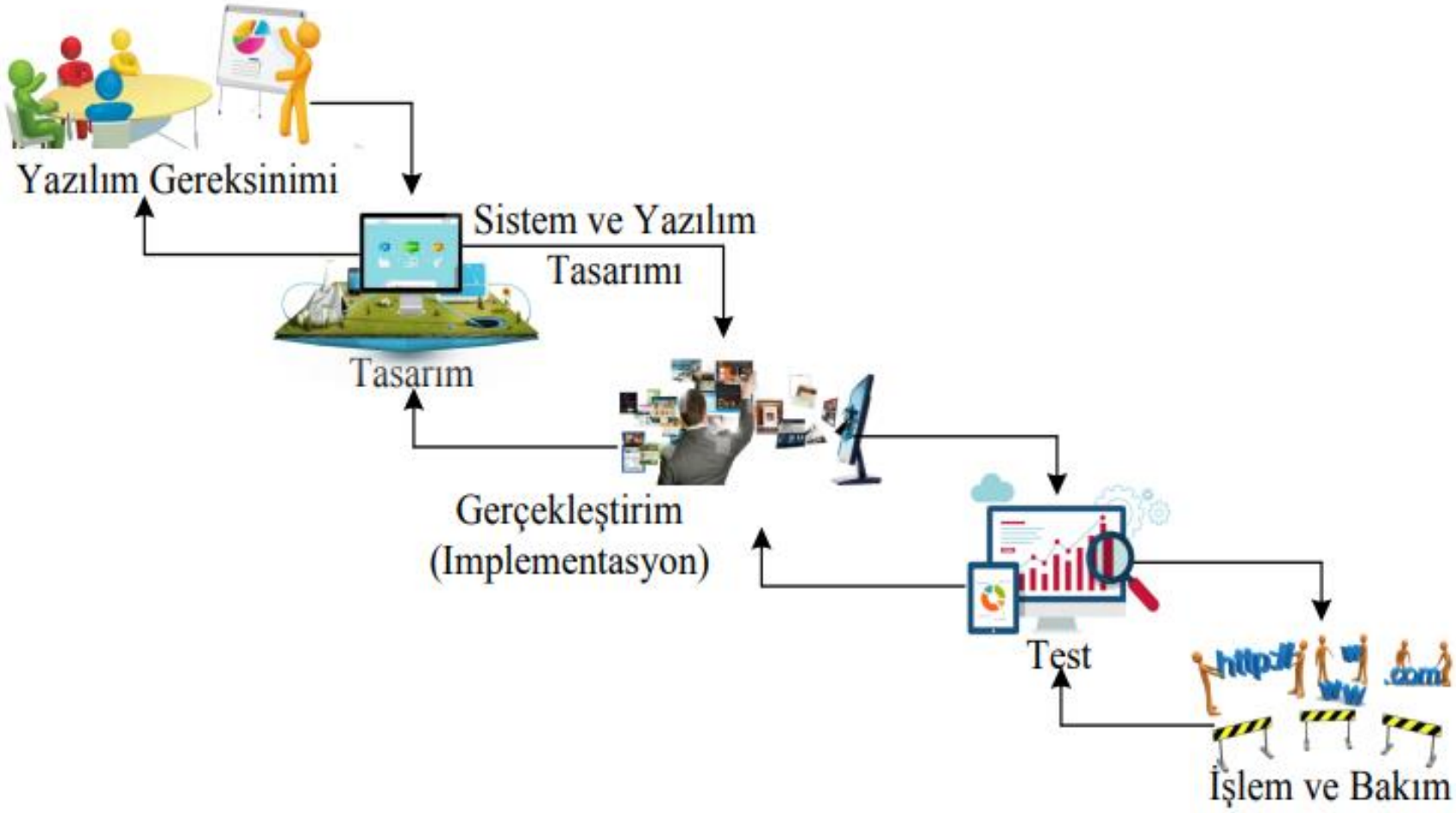
- Herhangi bir model ya da yöntem yok.
- Geliştiren kişiye bağımlı (belli bir süre sonra o kişi bile sistemi anlayamaz hale gelebilir ve geliştirme güçlüğü yaşar).
- İzlenebilirliği ve bakımı oldukça zor.
- 60’lı yıllarda kullanılan ilkel model
- Genellikle tek kişilik üretim ortamı.
- Basit programlama.

Barok Modeli



- Yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği model.
- 70'li yıllar.
- Belgelemeyi ayrı bir süreç olarak ele alır, ve yazılımın geliştirilmesi ve testinden hemen sonra yapılmasını 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 (Şelale) Modeli



Çağlayan (Şelale) 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.

Çağlayan (Şelale) Modeli

- 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.

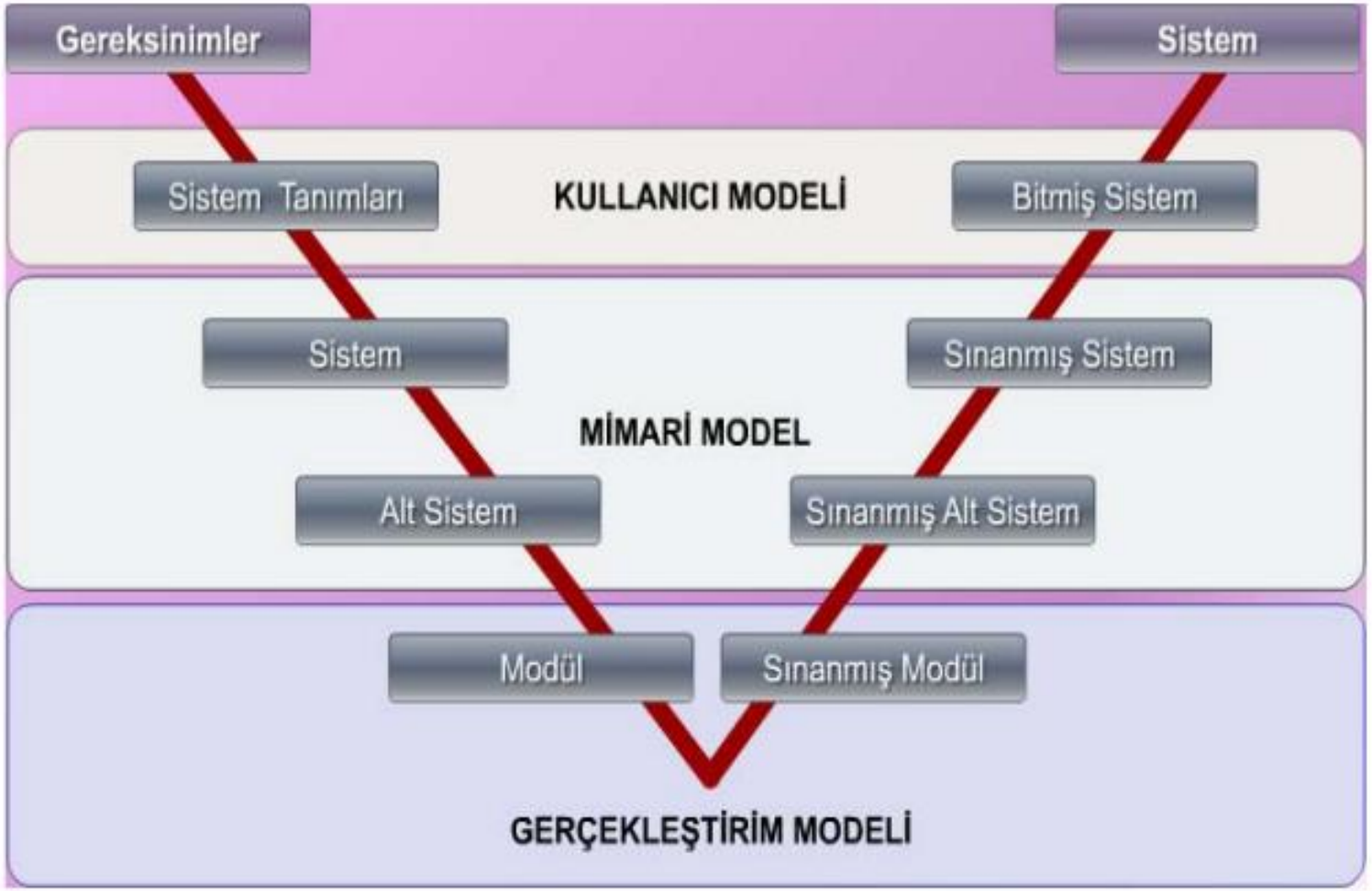
Avantajları

- Müşteriler ve son kullanıcılar tarafından da iyi bilinen 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. Aşamaları iyi anlaşılabilir.
- Proje yöneticileri için işin dağılımını yapma açısından kolaydır.
- Gereksinimleri iyi anlaşılabilen projelerde iyi çalışır.
- Kalite gereksinimlerinin bütçe ve zaman kısıtlamasına göre çok daha önemli olduğu projelerde iyi çalışır.

Sorunları

- Gerçek yaşamdaki projeler genelde yineleme gerektirir.
- Genelde yazılımın kullanıcıya ulaşma zamanı uzundur.
- Gereksinim tanımlamaları çoğu kez net bir şekilde yapılamadığından dolayı, yanlışların düzeltilme ve eksiklerin giderilme maliyetleri yüksektir.
- Yazılım üretim ekipleri bir an önce program yazma, çalıştırma ve sonucu görme eğiliminde olduklarından, bu model ile yapılan üretimlerde ekip mutsuzlaşmakta ve kod yazma dışında kalan (ve iş yükünün %80'ini içeren) kesime önem vermemektedirler.
- Üst düzey yönetimlerin ürünü görme süresinin uzun oluşu, projenin bitmeyeceği ve sürekli gider merkezi haline geldiği düşüncesini yaygınlaştırmaktadır.

V Süreç Modeli



V Süreç 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 alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler.

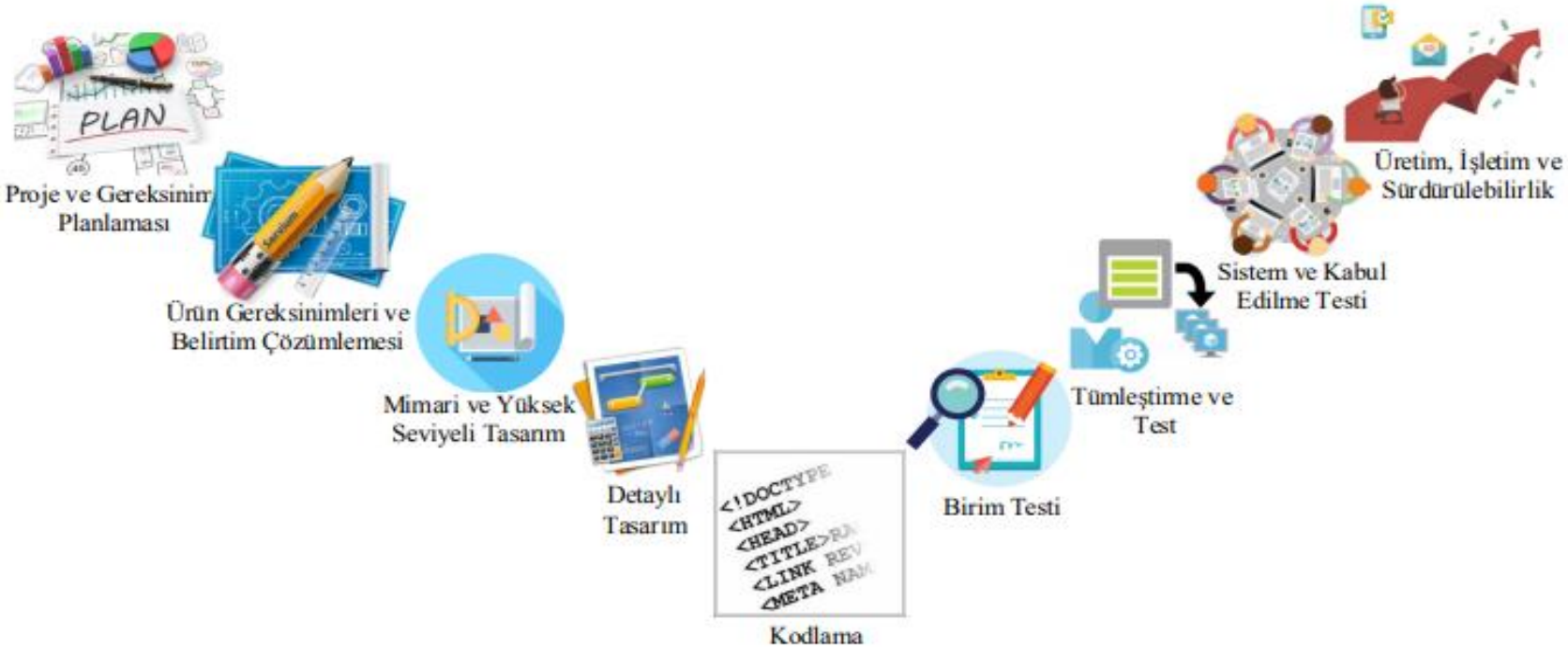
- Gerçekleştirim Modeli

  Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar.

V Süreç 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-Avantajlar



Verification ve validation planları erken aşamalarda vurgulanır.

Verification ve validation sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır.

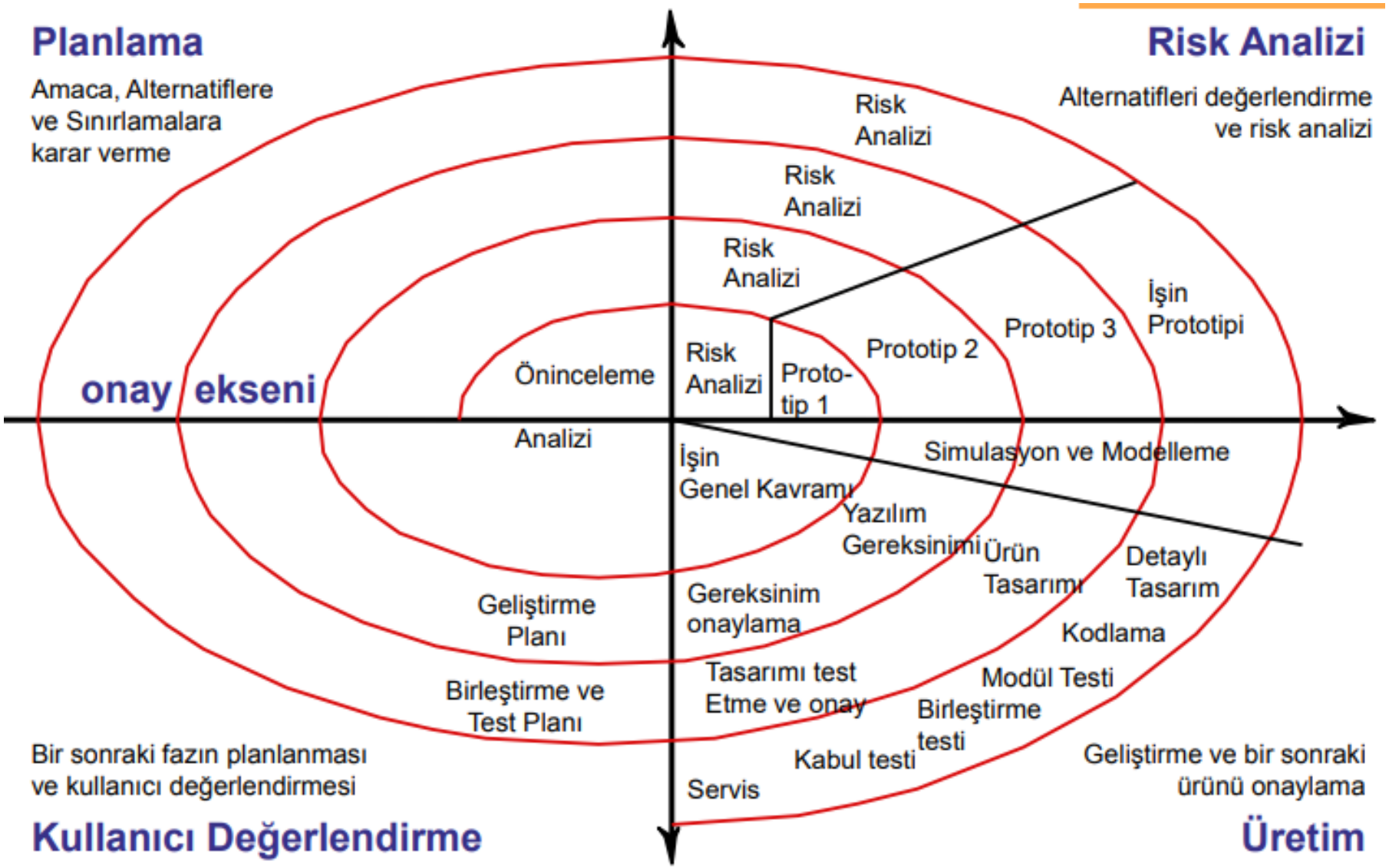
Proje yönetimi tarafında takibi kolaydır.

Kullanımı kolaydır.

V Modeli-Dezavantajlar

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

Helezonik(Spiral) Modeli



Helezonik Model

1. Planlama: Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirme

2. Risk Analizi: Risk seçeneklerinin araştırılması ve risklerin belirlenmesi
3. Üretim: Ara ürünün üretilmesi

4. **Kullanıcı Değerlendirmesi**

Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler

Helezonik modelin avantajları

1. Kullanıcı Katkısı

Üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır. Yazılımı kullanacak personelin sürece erken katılması ileride oluşabilecek istenmeyen durumları engeller.

2. Yönetici Bakışı

Gerek proje sahibi, gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla proje boyunca karşılaştıkları için daha kolay izleme ve hak ediş planlaması yapılır.

3. Yazılım Geliştirici (Mühendis) Bakışı

Yazılımın kodlanması ve sınanması daha erken başlar.

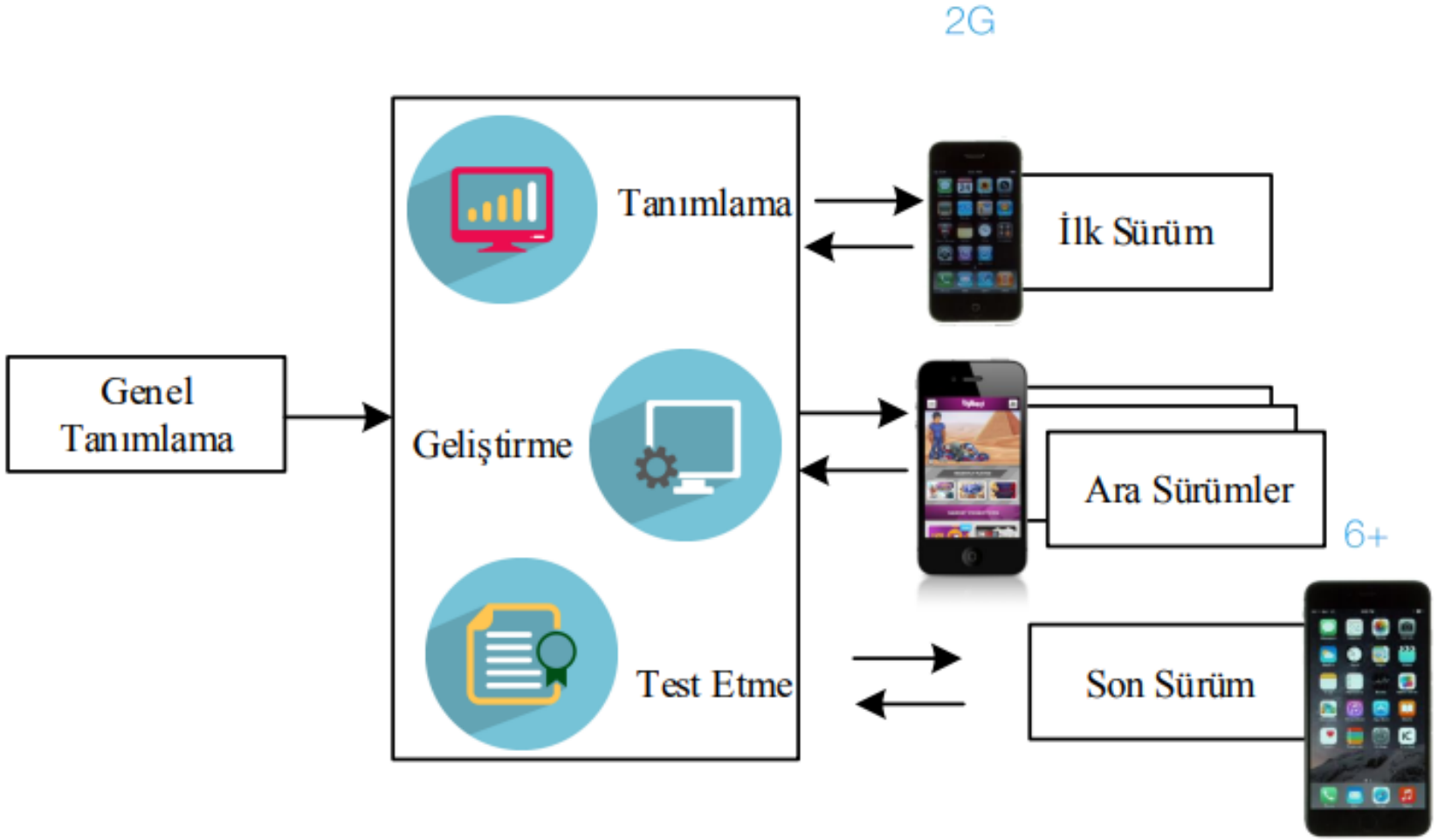
Helezonik Model

- Risk Analizi Olgusu ön plana çıkmıştır.
- Her döngü bir fazı ifade eder.
- Doğrudan tanımlama, tasarım,... vs gibi bir faz yoktur.
- Yinelemeli artımsal bir yaklaşım vardır.
- Prototip yaklaşımı vardır.

Evrimsel Geliştirme Süreç Modeli

- İlk tam ölçekli modeldir.
- 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 Süreç Modeli



Evrimsel Geliştirme Süreç 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

Aksaklığı

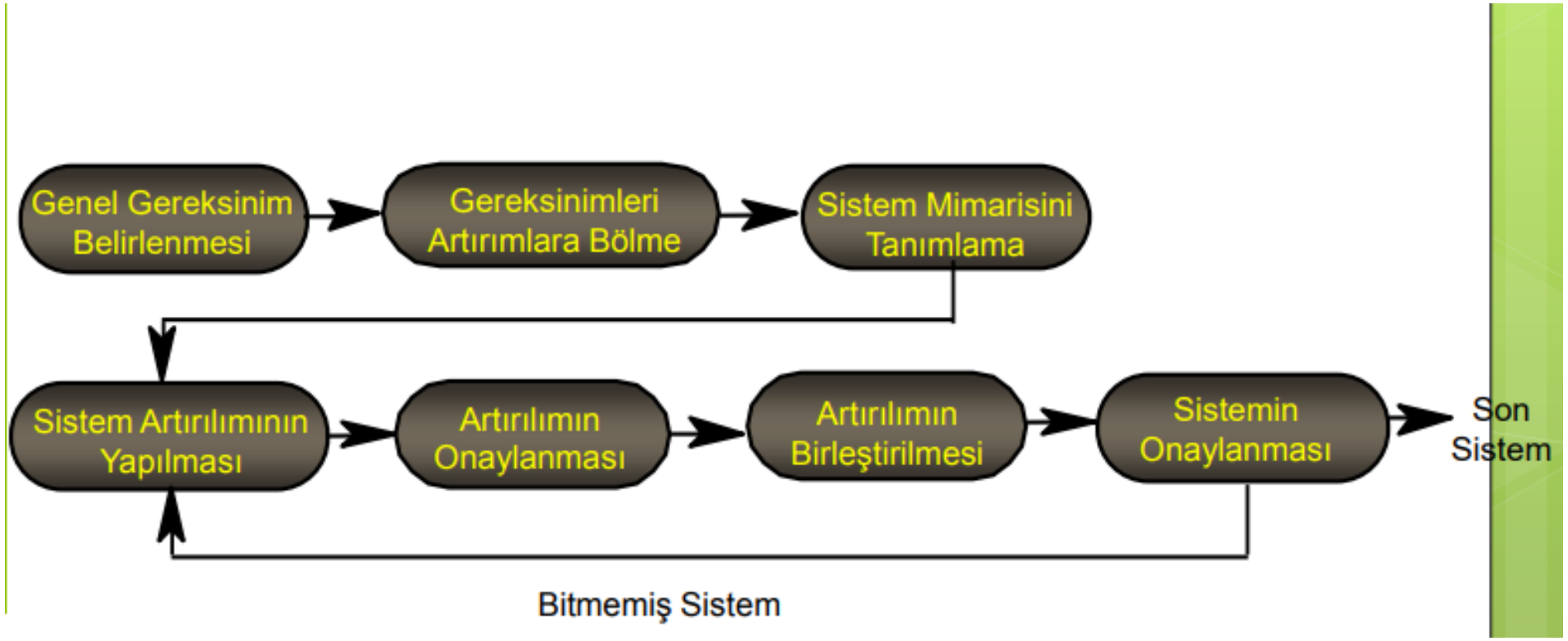
- Değişiklik denetimi
- Konfigürasyon Yönetimidir
  - Sürüm Yönetimi
  - Değişiklik Yönetimi
  - Kalite Yönetimi

Artırımsal Geliştirme Süreç Modeli

- 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 Süreç Modeli

Artırımsal Geliştirme Süreç Modeli



Araştırma Tabanlı Süreç Modeli

- 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!

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.