Yazılım Mühendisliği
Süreç Modelleri

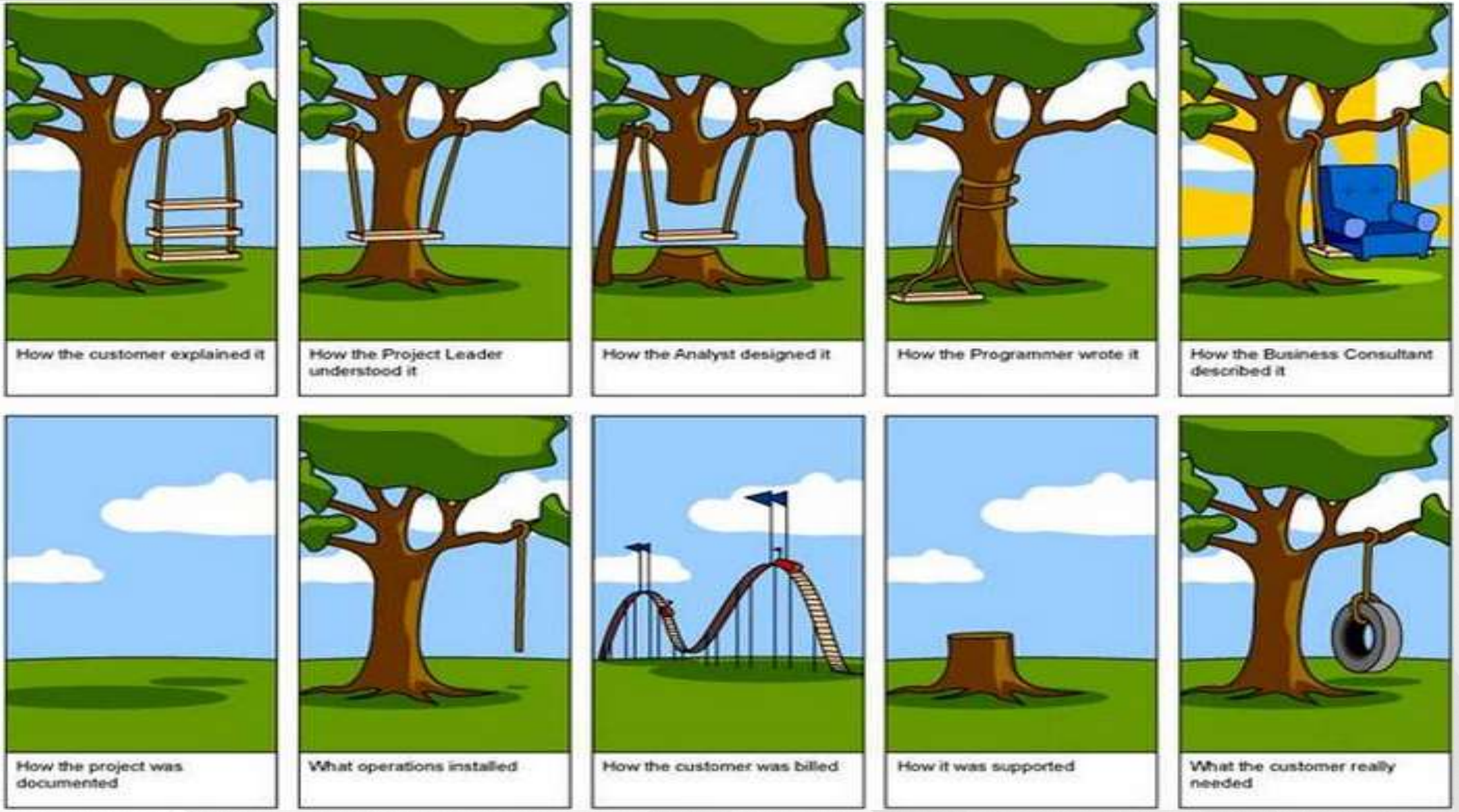
HEDEFLER

- ✓ Yazılım Yaşam Döngüsü (Çekirdek Süreçler)
- ✓ Belirtim Yöntemleri –Süreç Modelleri
- ✓ Yazılım Süreç Modelleri
- ✓ Gelişigüzel Model
- ✓ Barok Model
- ✓ Çağlayan Modeli
- ✓ V Modeli
- ✓ Spiral Model
- ✓ Evrimsel Geliştirme Süreç Modeli
- ✓ Artırımsal Geliştirme Süreç Modeli
- ✓ Araştırma Tabanlı Süreç Modeli
- ✓ Metodolojiler

Hedef

Yazılım geliştirme yaşam döngüsünün temel aşamaları olan,

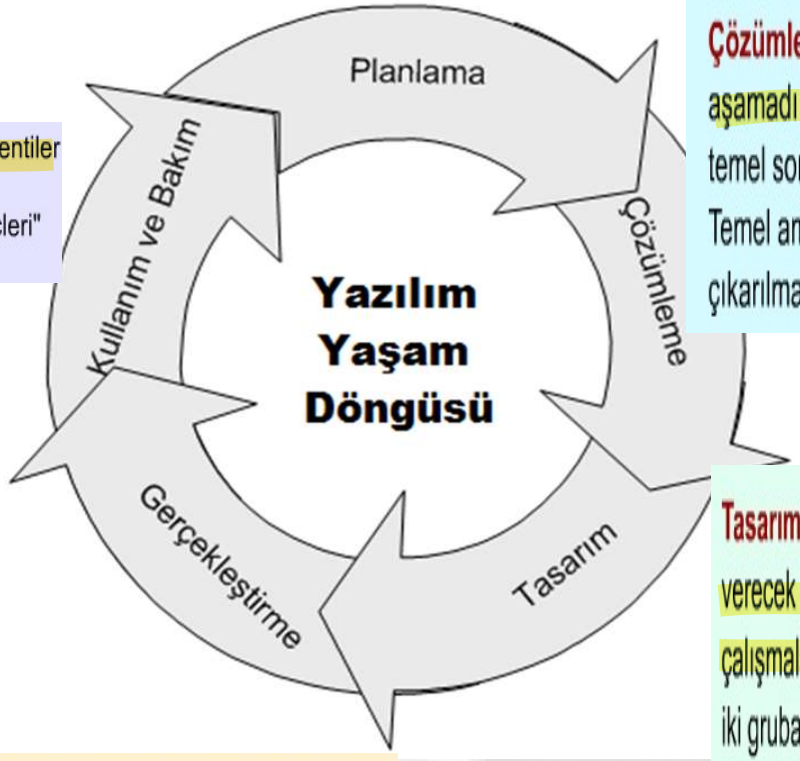
- ✓Planlama aşaması,
- ✓Çözümleme aşaması,
- ✓Tasarım aşaması,
- ✓Gerçekleştirim aşaması ve
- ✓Bakım aşaması anlatılarak, yaşam döngüsü modellerinin (süreç modelleri) açıklanması hedeflenmektedir.


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Yazılım Yaşam Döngüsü

Planlama: Üretilecek yazılım ile ilgili olarak, personel ve donanım gereksinimlerinin çıkarıldığı, olurluk çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.

Bakım: İşletime alınan yazılım ile ilgili olarak, hata giderme ve yeni eklentiler yapma aşamasıdır. Bu aşama yazılımın tüm yaşamı boyunca sürer. Yukarıda belirtilen adımlar, yazılım yaşam döngüsünün "çekirdek süreçleri" olarak tanımlanır.



Çözümleme: Yazılımın işlevleri ile gereksinimlerin ayrıntılı olarak çıkarıldığı aşamadır. Bu aşamada temel olarak mevcut sistemde varolan 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.

Tasarım: Çözümleme aşamasından sonra belirlenen gereksinimlere karşılık verecek yazılım ya da bilgi sisteminin temel yapısının oluşturulması çalışmalarıdır. Bu çalışmalar, mantıksal tasarım ve fiziksel tasarım olmak üzere iki gruba ayrılır. Mantıksal tasarımda, mevcut sistem değil, bu kez önerilen sistemin yapısı anlatılır. Olası örgütsel değişiklikler önerilir. Fiziksel tasarımda ise yazılımı içeren bileşenler ve bunların ayrıntıları içerilir.

Gerçekleştirim: Tasarım aşaması biten yazılımın kodlama, sınama ve kurma çalışmalarının yapıldığı aşamadır.

Yazılım Yaşam Döngüsü

1. Planlama

   Personel ve donanım gereksinimlerinin çıkarıldığı, fizibilite çalışmasının yapıldığı ve proje planının oluşturulduğu aşamadır.

2. Analiz

   Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak çıkarıldığı aşama. Var olan işler incelenir, temel sorunlar ortaya çıkarılır.

   mantıksal; önerilen sistemin yapısı anlatılır,

3. Tasarım

   Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır.

   mantıksal; önerilen sistemin yapısı anlatılır,

   fiziksel; yazılımı içeren bileşenler ve bunların ayrıntıları.

4. Gerçekleştirim

   Kodlama, test etme ve kurulum çalışmalarının yapıldığı aşamadır.

5. Bakım

   Hata giderme ve yeni eklentiler yapma aşaması.

Belirtim Yöntemleri

Bir çekirdek sürece ilişkin fonksiyonları yerine getirmek veya çekirdek süreçler arası geçişlerin belirtilmesinde kullanılan yöntemler, Belirtim Yöntemleri olarak adlandırılır.

yazılım yaşam döngüsündeki çekirdek süreçlerin geliştirme aşamasında hangi sırada uygulanacağını tanımlayan modellere Süreç Modelleri denir.

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 Süreç 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. Spiral Model
6. Evrimsel Model
7. Artırımsal Model
8. Araştırma Tabanlı Model

Gelişigüzel Model-**1960lar**

- ✓Herhangi bir model ya da yöntem yok.
- ✓Geliştiren kişiye bağlı.
- ✓İzlenebilirliği ve bakımı oldukça zor.
- ✓Genellikle tek kişilik üretim ortamı.
- ✓Karmaşık olmayan yazılım sistemleri.

Barok Model-**1970ler**

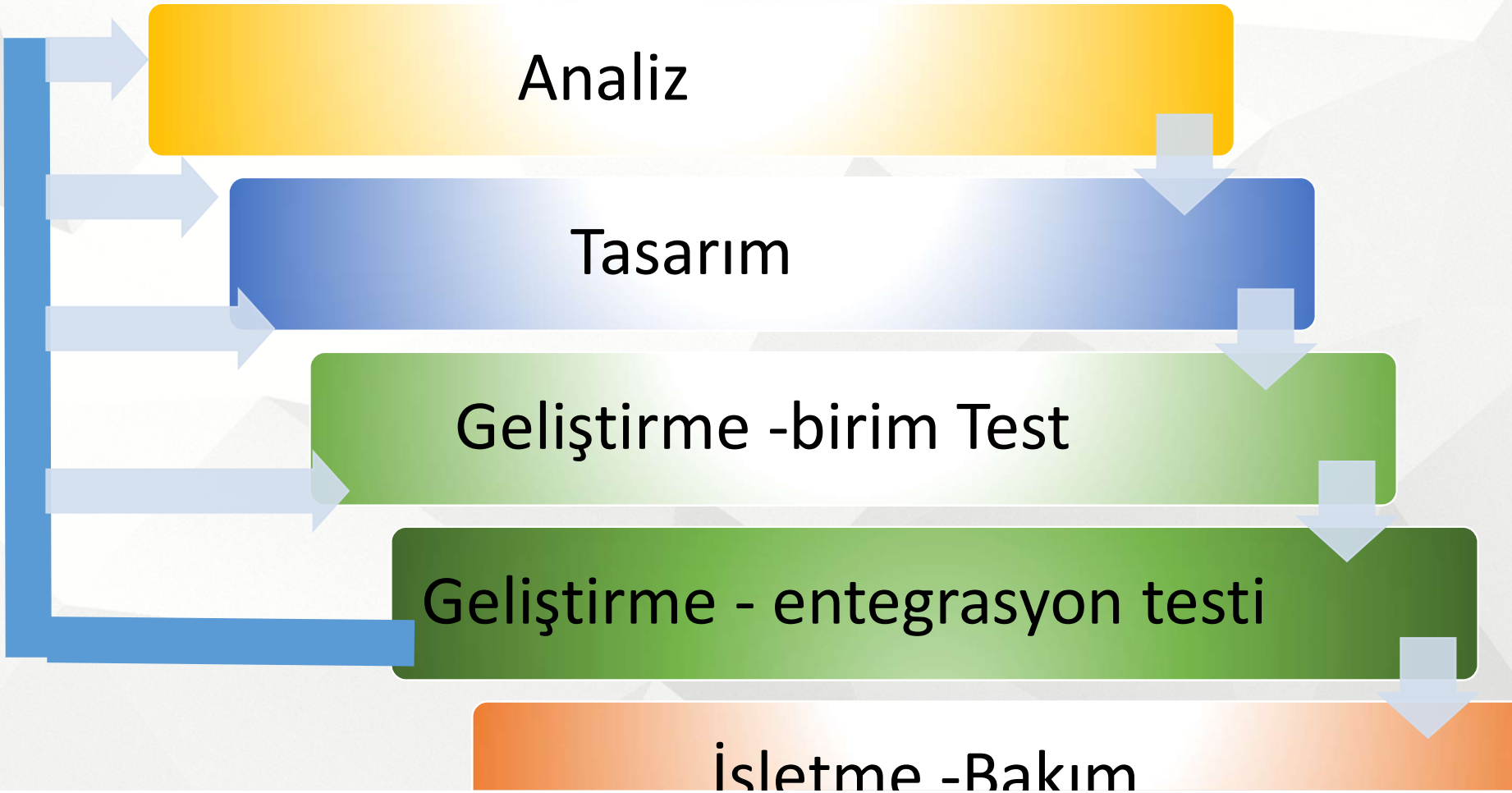
- ✓ Analiz
- ✓ Tasarım
- ✓ Kodlama
- ✓ Modül Testleri
- ✓ Altsistem Testleri
- ✓ Sistem Testi
- ✓ Belgeleme
- ✓ Kurulum

Yaşam döngüsü temel adımlarının doğrusal bir şekilde geliştirildiği model.

Belgelemeyi ayrı bir süreç olarak ele alır, ve yazılımın geliştirilmesi ve testinden hemen sonra yapılmasının öngörür.

Aşamalar arası geri dönüşlerin nasıl yapılacağı tanımlı değil.

Çağlayan Modeli- **(Klasik Model)**



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.

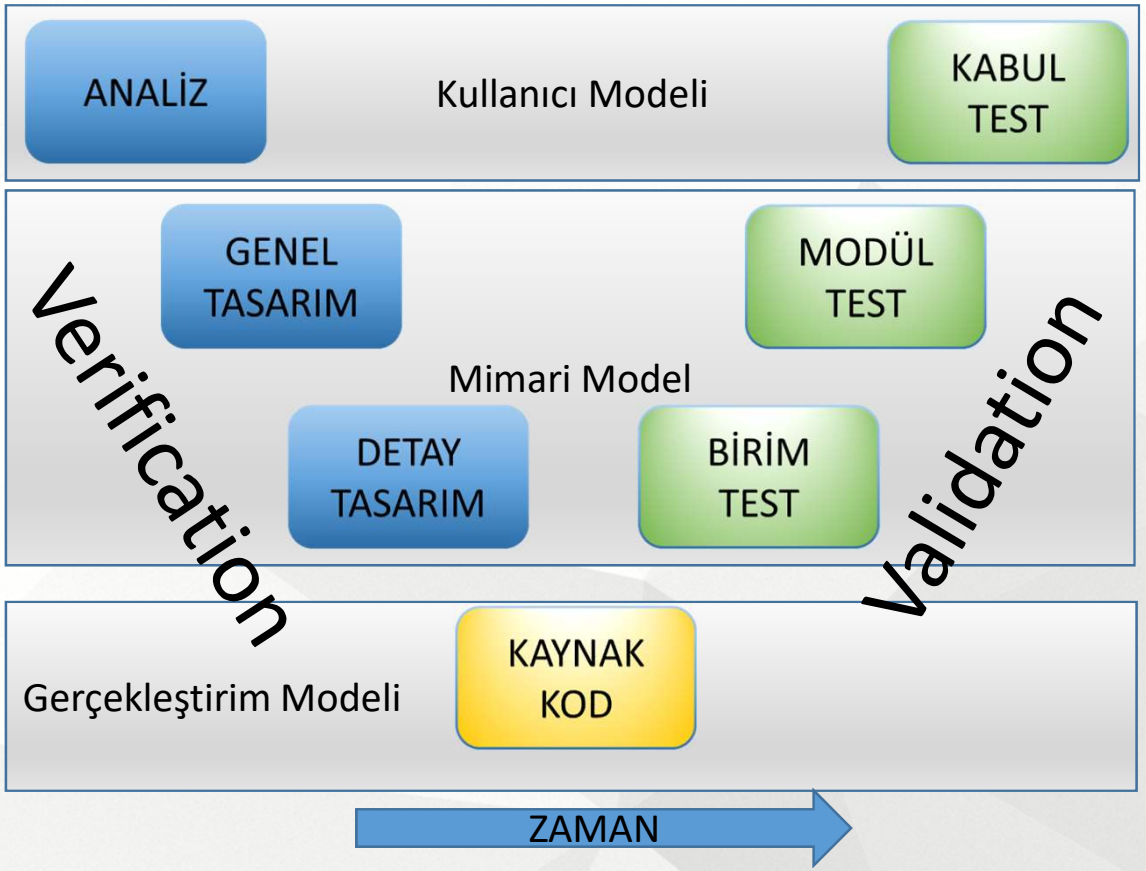
Çağlayan Modeli

- ✓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

Çağlayan Modeli –Zorluklar

- ✓Bir sonraki aşamaya geçmeden, önceki aşama neredeyse tümüyle tamamlanmış olmalıdır (örneğin, gereksinim tanımlama aşaması bitmeden tasarım aşamasına geçilemez.)
- ✓Bu şekilde geliştirme boyunca değişen müşteri isteklerinin sisteme yansıtılması zorlaşır.
- ✓Önceki nedenle bu model, gereksinimleri iyi tanımlı ve değişiklik oranı az olacak sistemler için daha uygundur.
- ✓Çok az sayıda iş sisteminin gereksinimleri başlangıçta iyi şekilde tanımlanabilir. Bu zorluğu aşmak için; gereksinim tanımlama aşamasından önce iş gereksinimlerinin anlaşılması ve tanımlanması faydalı olabilir.
- ✓Daha çok, geniş kapsamlı sistem mühendisliği projeleri için tercih edilir.

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 altsistem 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, gerçekleşmektedir.

V Modeli Çıktıları

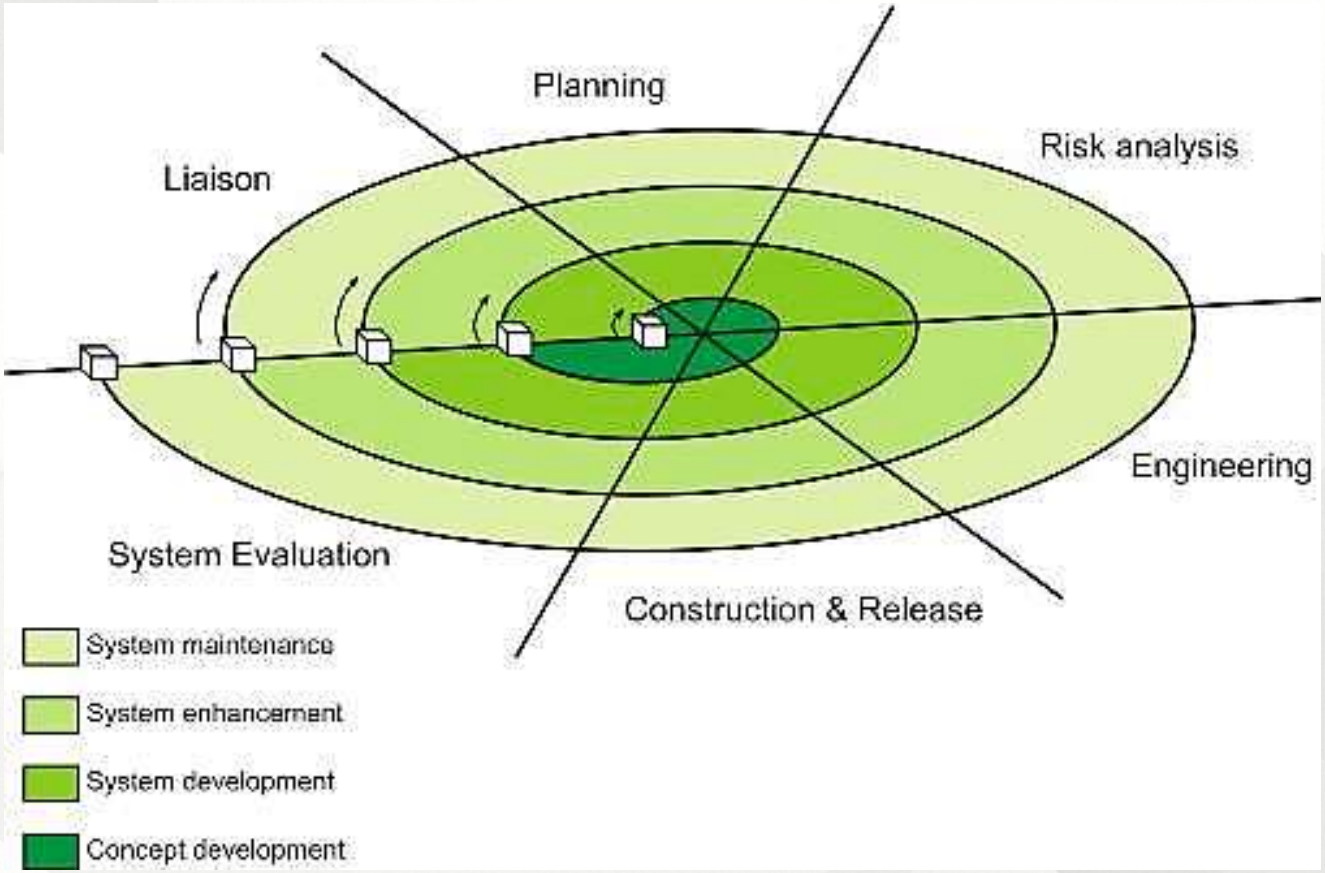
Kullanıcı Modeli	Mimari Model	Gerçekleştirim Modeli

Kullanıcı Modeli
Geliştirme sürecinin kullanıcı ile olan ilişkilerini tanımlamaktadır. Söz konusu ilişkiler, gerek üretim gerekse kabul (sınama) açısından kullanıcıdan beklenenleri ortaya koymaktadır. Üretim amacıyla, kullanıcı gereksinimlerini tanımlamakta ve bu yolla sistem tanımları ya da belirtimleri ortaya çıkarılmakta, öte yandan sistemin nasıl sınanacağı ya da kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.

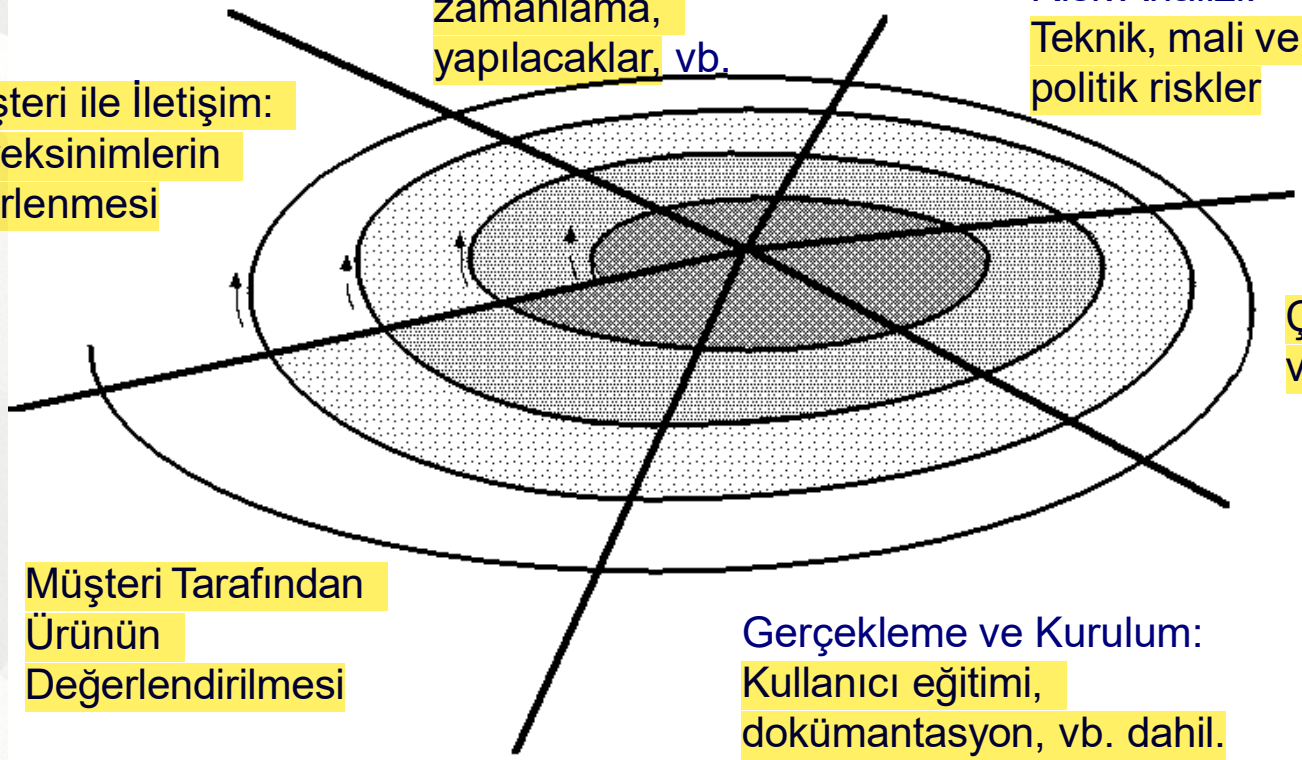
Mimari Model
Sistem tasarımı ve oluşacak alt sistem ve tüm sistemin sınama işlemlerine ilişkin işlevleri içermektedir.

Gerçekleştirim Modeli
Yazılım modüllerinin kodlanması ve sınanmasına ilişkin işlevleri içermektedir.

Spiral Model



SARMAL (Spiral) MODEL



Planlama: Kaynaklar, zamanlama, yapılacaklar, vb.

Risk Analizi: Teknik, mali ve politik riskler

Müşteri ile İletişim: Gereksinimlerin Belirlenmesi

Çözümleme ve Tasarım

Müşteri Tarafından Ürünün Değerlendirilmesi

Gerçekleme ve Kurulum: Kullanıcı eğitimi, dokümantasyon, vb. dahil.

Spiral Model

Tasarımı doğrusal bir süreç olarak gören diğer modellerin aksine, bu model spiral bir süreç olarak görür. Bu, yineleyici tasarım döngülerini genişleyen bir spiral olarak temsil ederek yapılır.

Genellikle iç çevrimler, gereksinim tanımının rafine edilmesi için prototipleme ile birlikte ihtiyaç analizinin erken evresini ve dış spiraller yazılım tasarımını aşamalı olarak temsil eder.

Her helezonda, tasarım çabalarını ve bu yineleme için ilgili riski değerlendirmek için bir risk değerlendirme aşaması vardır. Her spiralin sonunda, mevcut spiralin gözden geçirilebilmesi ve bir sonraki aşamanın planlanabilmesi için gözden geçirme aşaması vardır.

Spiral Model

Her tasarım sarmalının altı ana faaliyeti altı temel görevle temsil edilmektedir:

1. Müşteri İletişimi
2. Planlama
3. Risk Analizi
4. Yazılım Tasarımı
5. Üretim-dağıtım
6. Müşteri onayı

Avantajları

1. Risk analizi yapmaktadır.
2. Bu yazılım tasarım modeli, büyük yazılım projelerini tasarlamak ve yönetmek için daha uygundur.

Dezavantajları

1. Risk analizi yüksek uzmanlık gerektirir.
2. Kullanması pahalı model
3. Küçük projeler için uygun değildir.

Spiral Modelin Üstün Yönleri

Kullanıcı Katkısı

Helezonik modele göre oluşturulan metodolojiler üretim süreci boyunca ara ürün üretme ve üretilen ara ürünün kullanıcı tarafından sınanması temeline dayanır. Özellikle uzun projelerde kullanıcının erken dönemlerde sürecin içerisine katılması, sürekli geri bildirimler vermesi ve proje bitiminde sürprizlerle karşılaşılmaması gibi olgular üretim başarısını olumlu olarak etkiler.

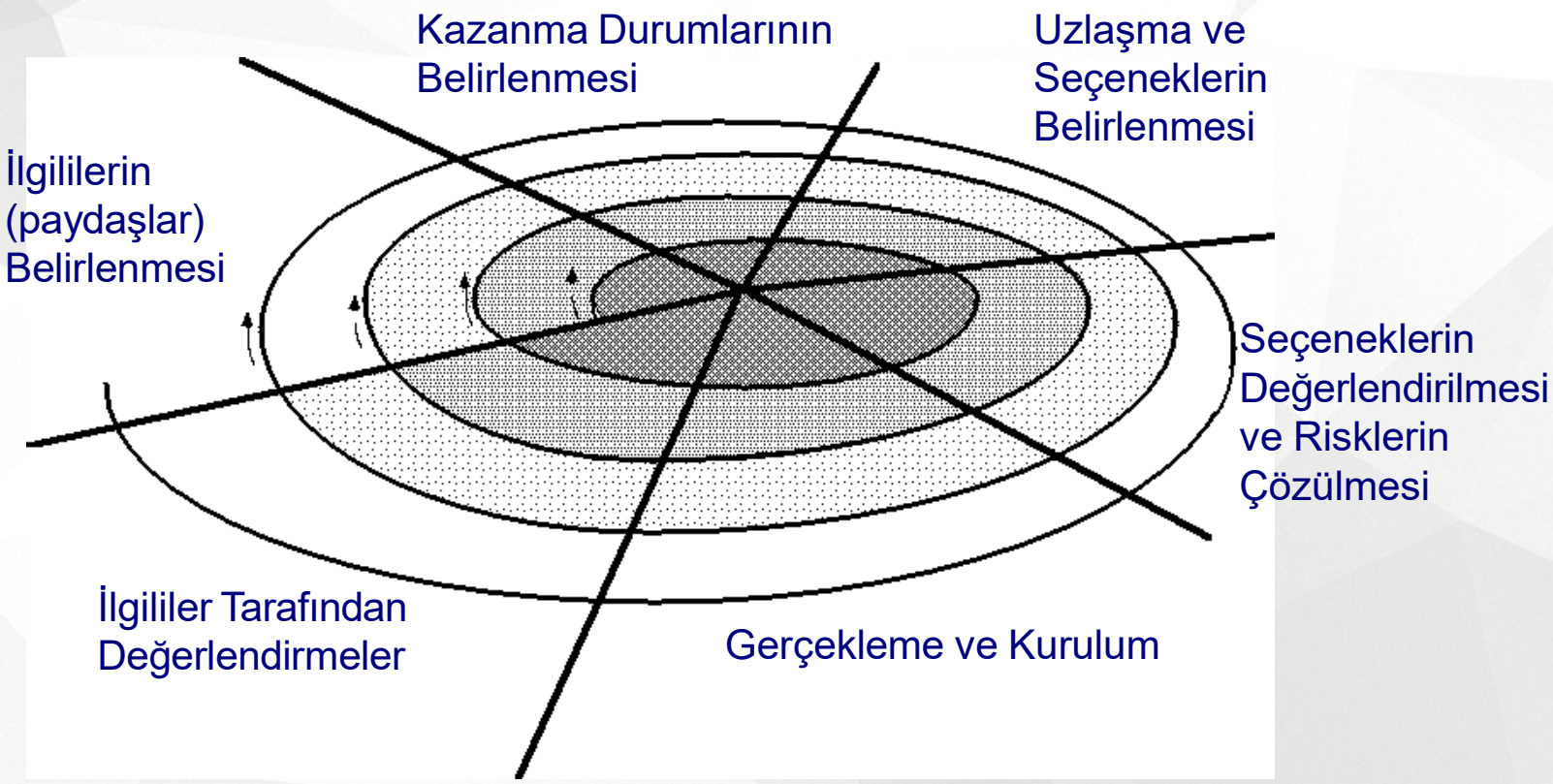
Yönetici Bakışı

Gerek proje sahibi gerekse yüklenici tarafındaki yöneticiler, çalışan yazılımlarla (ara ürün) proje süreci boyunca karşılaştıklarından dolayı daha kolay izleme ve hakediş planlama kolaylığı elde ederler.

Yazılım Geliştirici ya da Mühendisi Bakışı

Yazılımın kodlanma ve sınanması Çağlayan modeline oranla daha erken başlar.

Kazan-Kazan Sarmal Modeli (WINWIN Spiral Model)

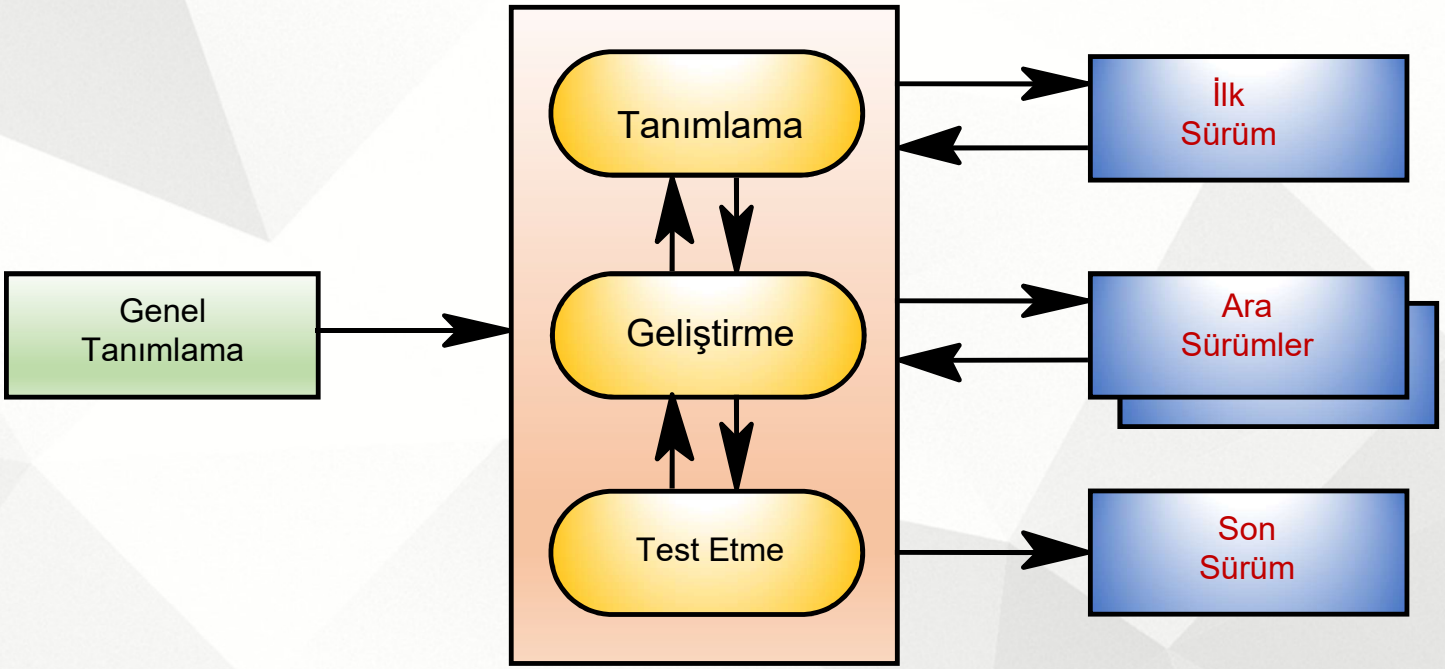


- Paydaş: Yazılımın başarısı ve başarısızlığının etkileyeceği kişi ve kurumlar.
- Eksiler : Her planlama aşamasında bütçenin yeniden değerlendirilmesi, sabit bütçe isteyen(!) yöneticileri mutlu etmez.

Evrimsel (“Evolutionary”) Geliştirme Süreç Modeli

- İlk tam ölçekli modeldir.
- Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir.
- 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

- ✓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.

EKSİKLERİ

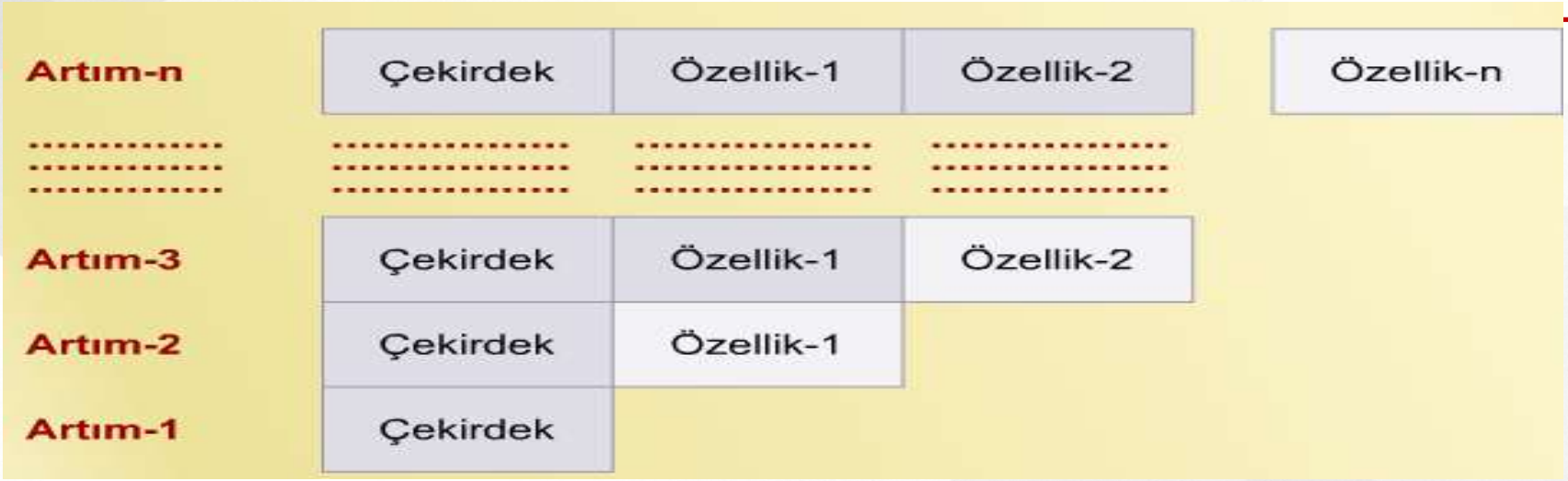
- ✓Değişiklik denetimi

- ✓Konfigürasyon Yönetimidir
  - Sürüm Yönetimi
  - Değişiklik Yönetimi
  - Kalite Yönetimi

Artırımsal Geliştirme Süreç Modeli

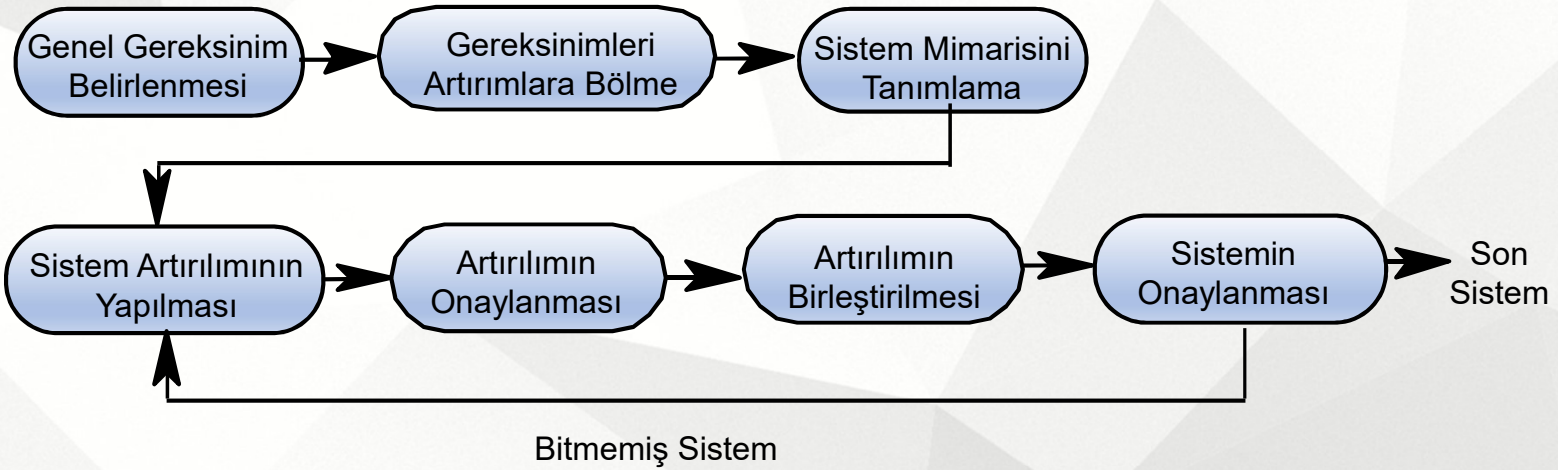
- ✓Üretilen her yazılım sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir.
- ✓Öğrencilerin bir dönem boyunca geliştirmeleri gereken bir programlama ödevinin 2 haftada bir gelişiminin izlenmesi (bitirme tezleri).
- ✓Uzun zaman alabilecek ve sistemin eksik işlevlikle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

Artımsal (“Incremental”) Geliştirme Süreç Modeli



Modelde üretilen ve uygulamaya alınan her ürün sürümü birbirini içerecek şekilde giderek artan sayıda işlev içerecek biçimde geliştirilmektedir. Öncelikle ürüne ilişkin çekirdek bir kısım geliştirilerek uygulamaya alınmakta ardından yeni işlevsellikler eklenerek yeni sürümler elde edilmektedir.

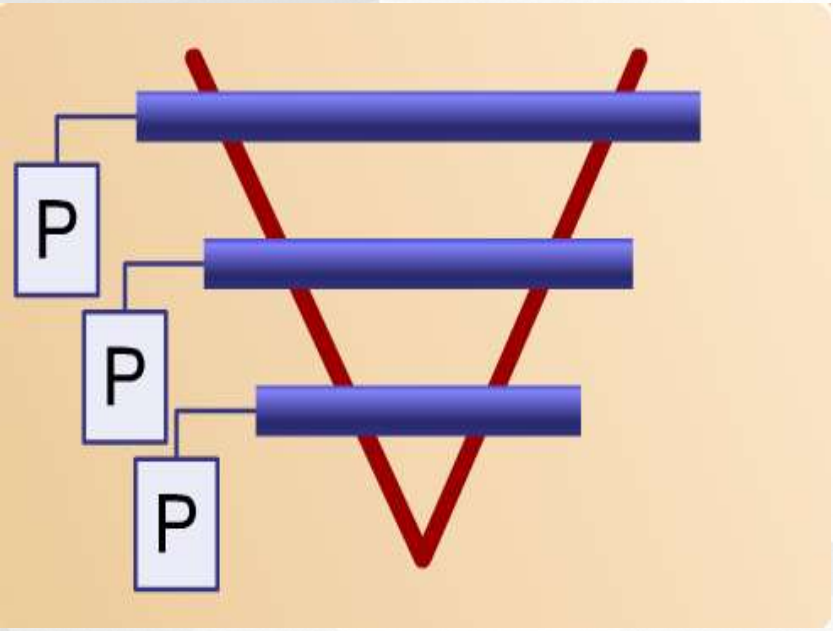
Artırımsal Geliştirme Süreç Modeli



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.

Prototipleme Modeli



Prototip geliştirmedeki iş adımları:

1. Belirsizliği tanımla
2. Çözümleri tanımla
3. Prototip çalışması yap
4. Belirsizliğin sonucunu elde et

Prototipleme modelinin temel zayıf yönü, kaynak maliyetlerinin kestirimindeki zayıflığıdır. Bir prototip çalışmasının ne kadar süreceği, ne kadar iş yükü gerektireceği kolayca kestirilememekte ve dolayısıyla yönetimi zorlaşmaktadır.

Prototipleme işlevinin temel amacı, ilgili alt modellerde ortaya çıkabilecek belirsizlikleri azaltmaktır. Prototip çalışması araştırma türü bir çalışma olabileceği gibi sonradan atılacak bir yazılım parçası da olabilir.

Metodolojiler

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

Bir metodoloji,

- bir süreç modelini ve
- belirli sayıda belirtim yöntemini içerir

Günümüzdeki metodolojiler; çağlayan, V ya da spiral süreç modellerini temel almaktadır.

Bir Metodolojide Bulunması Gereken Temel Bileşenler

- ✓ Ayrıntılandırılmış bir süreç modeli
- ✓ Ayrıntılı süreç tanımları
- ✓ İyi tanımlı üretim yöntemleri
- ✓ Süreçler arası ara yüz 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 Metodolojide Bulunması Gereken Temel Bileşenler

- ✓Metodoloji bileşenleri ile ilgili olarak bağımsız kuruluş (IEEE, ISO, vs.) ve kişiler tarafından geliştirilmiş çeşitli standartlar ve rehberler mevcuttur.

- ✓Kullanılan süreç modelleri ve belirtim yöntemleri zaman içinde değiştiği için standart ve rehberler de sürekli güncellenmektedir.

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

HIZLI UYGULAMA GELİŞTİRME (RAD: Rapid Application Development)

- Kısa geliştirme çevrimleri üzerinde duran artımsal bir model.
- Gereksinimler:
  - 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 hıza uyum sağlayabilmelidirler.
  - Yüksek teknik risklere uygun değil.
- Sonuç:
  - Prototip geliştirmede kullanılması veya ana fikirlerinin diğer süreçlere uygulanması yerinde olacaktır.

BİLEŞEN TABANLI (Component Based) UYGULAMA GELİŞTİRME

- Uygulamanın hazır yazılım bileşenlerinden oluşturulmasını öngörür.

Lego parçaları gibi

- 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
  - Bileşenlerin uyarlanması gerekliliği
- Sonuçlar:
  - Özellikle hızlı uygulama geliştirme olmak üzere, ana fikirleri çeşitli süreçlere uygulanabilir.

ÇEVİK (Agile) SÜREÇLER

⑩ 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 çalışan yazılımdır.

ÇEVİK (Agile) SÜREÇLER

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

- Planlama:
  - Müşteri, kullanıcı öyküleri (KÖ) oluşturur.
    - Müşteri, öyküleri önemine göre derecelendirir.
    - Yaklaşık 3 haftada gerçeklenemeyecek 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 gerçeklenir,
    - Ya da önce yüksek öncelikli öyküler gerçeklenir.
    - Her olasılıkta tüm öyküler kısa sürede (birkaç hafta)

      ✓ gerçeklenmelidir.

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-Resposibility-Collaboration) kartları ile yazılımın sınıf düzeyinde incelenmesi.
  - Karmaşık bir tasarımdan kaçınılamazsa işlevsel bir ön gerçekleme yapılır (Spike solution).
  - Refactoring teşvik edilir.
  - Bu aşamanın ürünleri CRC kartları ve ön gerçeklemelerdir (başka ürün yok).

ÇEVİK (Agile) SÜREÇLER

- Sürü (Scrum)
  - Adımlar: Görev Listesi – Koşu – İşlev Gösterimi

- Görev Listesi:
  - Müşterinin belirlediği, kendisine değer sağlayan gereksinimler ve
    - ✓ özellikler (kullanıcı öykülerinin bir başka tanımı).
  - Önceliklendirilmiştir.

- Koşu:
  - Görev listesinin maddelerinden biri seçilir ve önceden belirlenmiş kısa bir
    - ✓ süre içerisinde (30 gün) gerçeklenir.
  - Koşu süresince ekibin her gün yaptığı toplantılar:
    - Kısa sürer (15dk).
    - Sürü önderi yönetir.
    - Herkesin sorduğu ve cevapladığı üç ana soru:
      - Son toplantıdan bu yana ne yaptınız?
      - Karşılaştığınız engeller nelerdir?
      - Yarınki toplantıda neleri başarmayı hedefliyorsunuz?