SİSTEM ANALİZİ VE TASARIMI

BİLGİ SİSTEMİ GELİŞTİRME SÜRECİ

Sistem Geliştirme Süreci ve Modelleri

Sistem Geliştirme Yaşam Döngüsü

- Bilgi sistemlerinin geliştirilmesi için izlenen sürece Sistem Geliştirme Yaşam Döngüsü denir.

ADIM	İŞLEM	ÇIKTILAR
Problemin Tanımı	Problemi ortaya koymak	İhtiyaçlar belirlenir
Fizibilite çalışması	Projenin kapsamı ve hedefleri ortaya konarak olabilirliğini belirlemek	Fizibilite çalışması raporu
Analiz	Problemin çözümlerini ortaya koymak	Çözümün lojik modeli
Genel Tasarım	Sistemin nasıl gerçekleştirileceğini belirleme	Sistemin maliyeti ve üst düzey tasarımı
Ayrıntılı Tasarım	Genel tasarımda belirlenen sisteme ait alt sistemlerin tanımlanması	Sistemin özellikleri ve ayrıntılı tasarım
Gerçekleştirme	Programı yazma, yükleme ve sınama	Çalışan sistem ve dokümantasyon
Bakım	Sistemin bakımını yaparak desteklemek	Çalışan Sistem

Geliştirme Süreç Modelleri

- Sistem geliştirmenin bahsedilen zorluklarıyla baş edebilmek için, geliştirmeyi sistematik hale getirmeyi hedefleyen çeşitli süreç modelleri ortaya çıkmıştır.
  - Bu modellerin temel hedefi; proje başarısı için, sistem geliştirme yaşam döngüsü ("system development life cycle") boyunca izlenmesi önerilen mühendislik süreçlerini tanımlamaktır.
  - Sistem geliştirme yaşam döngüsü: Bir sistemin ihtiyacının ortaya çıkmasından kullanımdan kalkmasına kadar geçen dönemdir.
- Modellerin ortaya çıkmasında, ilgili dönemin donanım ve yazılım teknolojileri ile sektör ihtiyaçları önemli rol oynamıştır.
  - Örnek:
  - Geleneksel modeller (örneğin; çağlayan ("waterfall") modeli)
  - Çevik ("agile") modeller (örneğin; uçdeğer ("extreme") programlama modeli -- XP)

Süreç ve Süreç Modeli

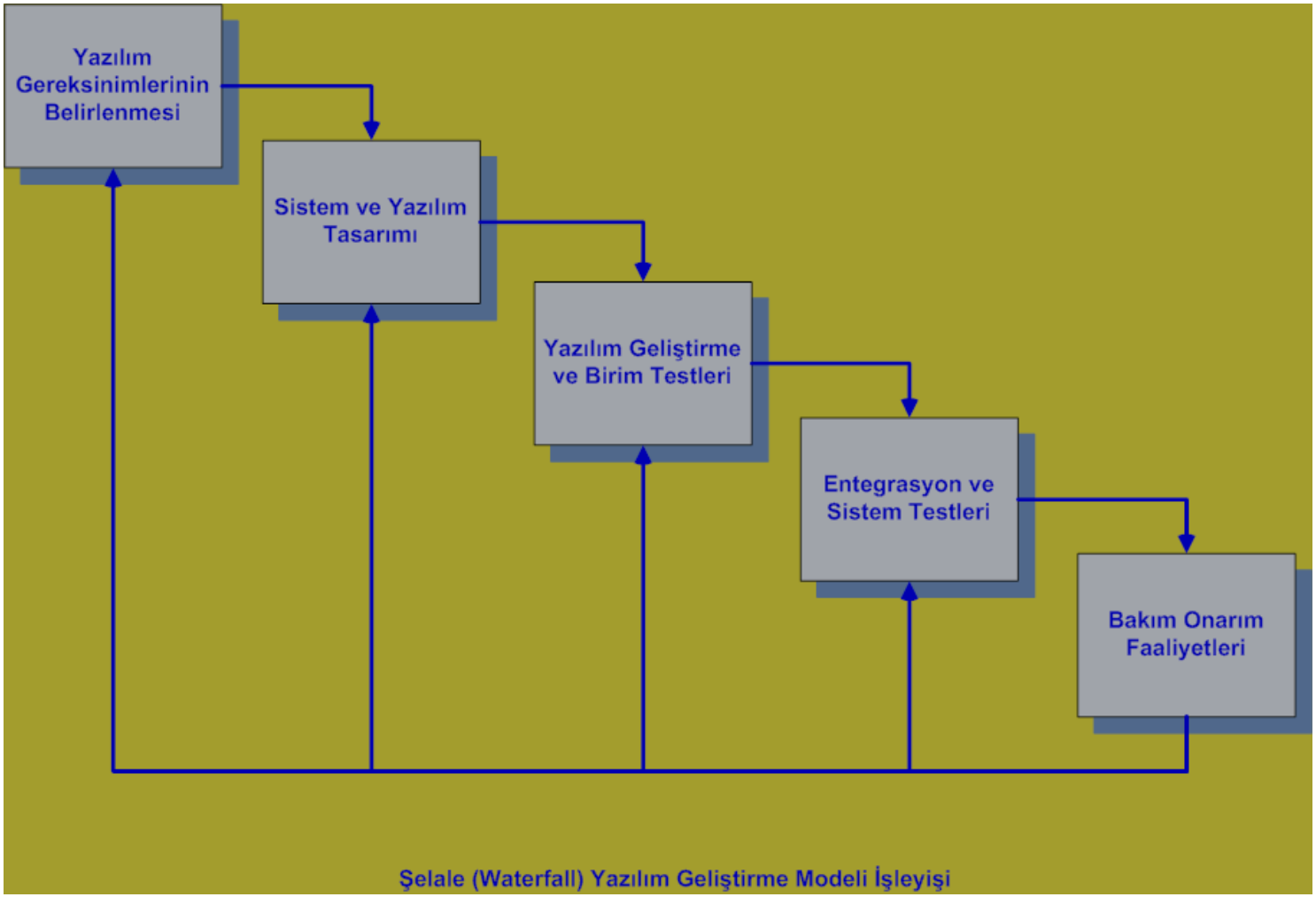
- Süreç nedir?
  - Belirli bir hedef için gerçekleştirilen adımlar zinciridir. [IEEE]
  - Sistemi ve ilişkili ürünlerini geliştirmek ve idame ettirmek için kullanılan etkinlikler, yöntemler, pratikler ve dönüşümlerdir.
  - Sistemi geliştirme ve idame amacı güden etkinlikler setidir.
- Süreç modeli nedir?
  - Bir geliştirme sürecinin belirli bir bakış açısıyla gösterilmiş, basitleştirilmiş temsilidir.
- Örnek bakış açıları:
  - İş-akışı → etkinlikler nasıl sıralı?
  - Veri-akış → bilgiler nasıl sıralı?
  - Rol-hareket → kim ne yapıyor?

Geleneksel Süreç Modelleri

- Çağlayan (“waterfall”) modeli
- Evrimsel (“evolutionary”) model
- Bileşen-tabanlı (“component-based”) model
- Artırımlı (“incremental”) model
- Döngüsel (“spiral”) model

Çağlayan Modeli – Aşamalar

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



Şelale (Waterfall) Yazılım Geliştirme Modeli İşleyişi

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 geliştirme projeleri için tercih edilir.

Evrimsel ("Evolutionary") Model

- Sistem, zaman içinde kazanılan anlayışa göre gelişir.
  - Amaç, müşteriyle birlikte çalışarak taslak bir sistem gereksinimleri tanımından çalışan bir sisteme ulaşmaktır.
- En iyi bilinen gereksinimlerle başlanır ve müşteri tarafından talep edildikçe yeni özellikler eklenir.
- Öğrenme amacıyla, sonradan atılabilecek prototipler ("throw-away prototyping") geliştirilir.
- Amaç, sistem gereksinimlerini anlamaktır.
- En az bilinen gereksinimlerle başlanır ve gerçek ihtiyaç anlaşılmaya çalışılır.

Evrimsel Model – Zorluklar

- Geliştirme süreci izlenebilir değildir. Her seferinde eklemelerle çalışan sistem, müşteriyle gözden geçirilir.
- Zaman içinde kazanılan anlayışa göre geliştirilen sistemler, sıklıkla kötü tasarlanır.
- Küçük- ve orta-ölçekli, etkileşimli ("interactive") sistemler için uygulanabilir.

Evrimsel Model

Artırımlı ("Incremental") Model

- Sistemi tek seferde teslim etmek yerine, geliştirme ve teslim parçalara bölünür. Her teslim beklenen işlevselliğin bir parçasını karşılar.
- Kullanıcı gereksinimleri önceliklendirilir ve öncelikli gereksinimler erken teslimlere dahil edilir.
- Bir parçanın geliştirmesi başladığında, gereksinimleri dondurulur. Olası değişiklikler sonraki teslimlerde ele alınır.

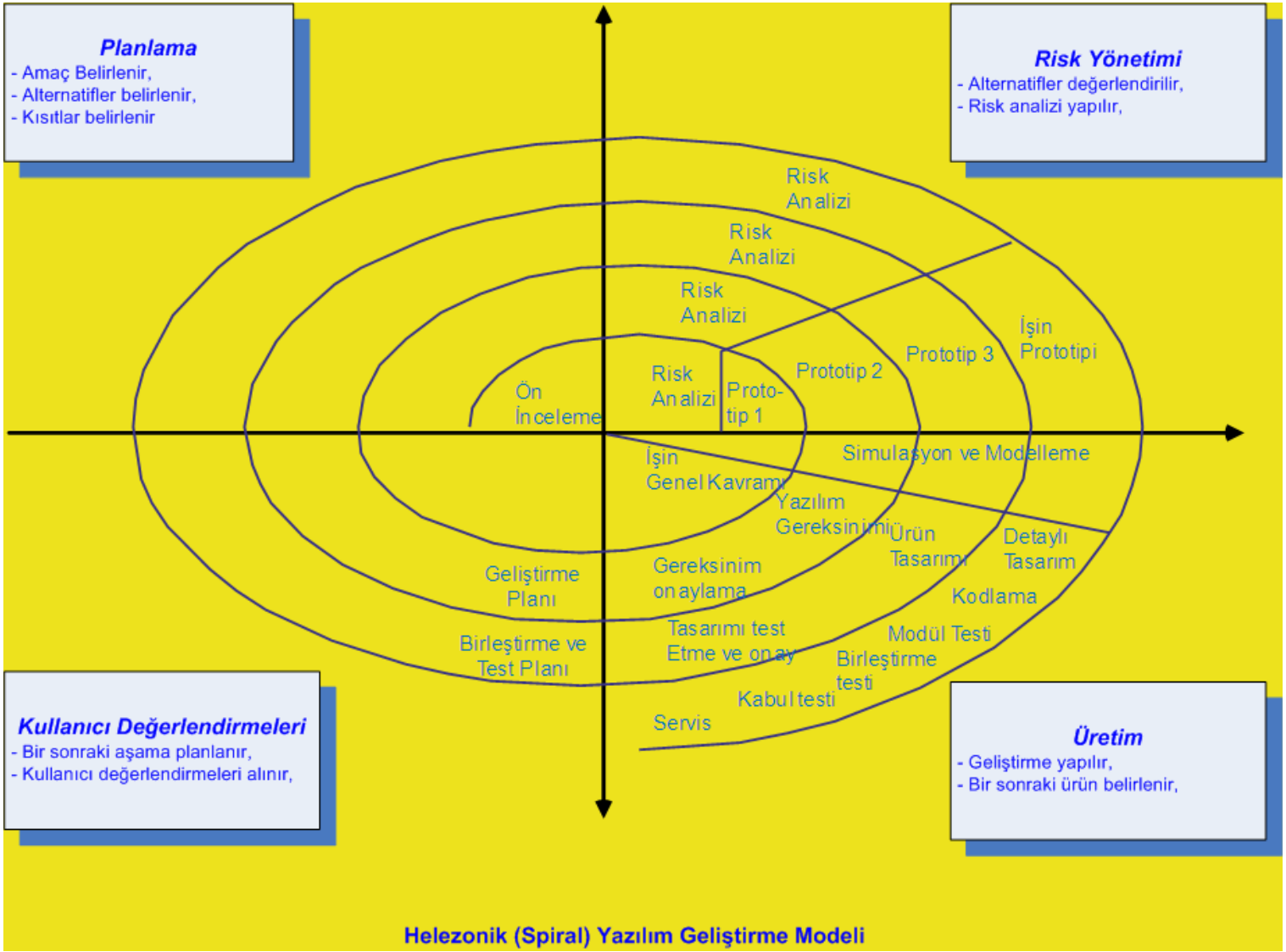
Artırımlı Model – Kazançlar

- Her teslimle birlikte müşteriye görünen bir değer döndüğünden, sistemin işlevselliği erken aşamalarda ortaya çıkar.
- Erken teslimler, sonraki teslimler için gereksinimleri çıkarmada prototip vazifesi görür.
- Projenin tümden batması riskini azaltır.
- Öncelikli gereksinimleri karşılayan sistem işlevleri daha çok test edilir.

Evrimsel Model

Döngüsel ("Spiral") Model

- Süreç, geri dönüşümlü etkinlikler zinciri yerine döngüsel olarak ifade edilir.
- Her döngü, süreçteki bir aşamayı ifade eder.
- 4 sektörden oluşur:
  - Hedef belirleme: Aşamanın başarımı için somut hedefler belirlenir.
  - Risk değerlendirme ve azaltma: Riskler adreslenerek azaltıcı eylemler gerçekleştirilir.
  - Geliştirme ve doğrulama: Genel modeller içinden geliştirme için bir model seçilir.
  - Planlama: Proje gözden geçirilir ve bir sonraki aşama planlanır.
- Riskler süreç boyunca özel olarak ele alınır ve çözümlenir.
- Tanımlama ve tasarım gibi sabit aşamalar yoktur; her döngü ihtiyaca göre seçilir.



Helezonik (Spiral) Yazılım Geliştirme Modeli

Çevik (“Agile”) Yazılım Süreç Modelleri

- Çevik modeller, mevcut geleneksel modellere alternatif olarak, 1990’larda ortaya çıkmaya başlamıştır.
  - Çevik: *Kolaylık ve çabuklukla davranan, tetik, atik.* [TDK]
  - 1950’lerdeki üretim alanında verimliliğin artırılması için geliştirilen yalın yaklaşımların, yazılım sektöründe bir uzantısı olarak ortaya çıkmıştır.
- Çevik modeller kapsamında; yazılım sistemlerini etkili ve verimli bir şekilde modellemeye ve belgelendirmeye yönelik, pratiğe dayalı yöntemler yer alır.
- Bu modelleme biçiminin; kapsadığı değerler, prensipler ve pratikler sayesinde geleneksel modelleme metotlarına göre yazılımlara daha esnek ve kullanışlı biçimde uygulanabileceği savunulmaktadır.

Çevik Yazılım Geliştirme Manifestosu

- 2001 yılında, dünyanın önde gelen çevik modellerinin temsilcileri, ortak bir zeminde buluşabilmek adına bir araya gelerek "çevik yazılım geliştirme manifestosu" nu yayınlamışlardır. Bu manifestoya göre;

  - Bireyler ve aralarındaki etkileşim, kullanılan araç ve süreçlerden;
  - Çalışan yazılım, detaylı belgelerden;
  - Müşteri ile işbirliği, sözleşmedeki kesin kurallardan;
  - Değişikliklere uyum sağlayabilmek, mevcut planı takip etmekten;

  daha önemli ve önceliklidir.

Çevik Yazılım Geliştirme Prensipleri

- Çevik yazılım geliştirme manifestosu maddelerinin altında yatan temel prensipler şunlardır:
  - İlk öncelik, sürekli, kaliteli yazılım teslimatıyla müşteri memnuniyetini sağlamaktır.
  - Proje ne kadar ilerlemiş olursa olsun değişiklikler kabul edilir. Çevik yazılım süreçleri değişiklikleri müşteri avantajına dönüştürür.
  - Mümkün olduğunca kısa zaman aralıklarıyla (2–6 hafta arası) çalışan, kaliteli yazılım teslimatı yapılır.
  - Analistler, uzmanlar, yazılımcılar, testçiler, vs. tüm ekip elemanları bire bir iletişim halinde, günlük olarak birlikte çalışır.
  - İyi projeler motivasyonu yüksek bireyler etrafında kurulur. Ekip elemanlarına gerekli destek verilmeli, ihtiyaçları karşılanarak proje ile ilgili tam güvenilmelidir.
  - Ekip içerisinde kaliteli bilgi akışı için yüz yüze iletişim önemlidir.
  - Çalışan yazılım, projenin ilk gelişim ölçütüdür.
  - Çevik süreçler mümkün olduğunca sabit hızlı, sürdürülebilir geliştirmeye önem verir.
  - Sağlam teknik altyapı ve tasarım çevikliği arttırır.
  - Basitlik önemlidir.
  - En iyi mimariler, gereksinimler ve tasarımlar, kendini organize edebilen ekipler tarafından yaratılır.
  - Düzenli aralıklarla ekip kendi yöntemlerini gözden geçirerek verimliliği arttırmak için gerekli iyileştirmeleri yapar.

Çevik Yazılım Süreç Modelleri

- Uçdeğer Programlama (“Extreme Programming – XP”)
- Scrum
- Özellik Güdümlü Geliştirme (“Feature-Driven Development – FDD”)
- Çevik Tümleşik Süreç (“Agile Unified Process – AUP”)

Uçdeğer Programlama - Prensipler (1)

- Basitlik Prensibi
  - Basit bir tasarım üzerinde odaklanmak, müşterinin onaylamayacağı ve uzun zaman alacak bir tasarımın yapılması riskini azaltır.
  - Kodu basit tutmak, gereksinimlerdeki değişikliklerin daha kolay yapılmasını sağlar. Doğru ve basit kodu yazan yazılımcı daha sonra iyileştirmeye zaman ayırır. Bu sayede değişecek müşteri isteklerine de zaman ayırır.
- İletişim Prensibi
  - Yazılım geliştirmenin yazılan raporlar ile yapılamayacağını, özellikle sözlü olmak üzere insanlar arasındaki birebir iletişimin her zaman daha yararlı olduğunu savunur. Belgeleme başka sebeplerden dolayı gerekli olabilir, ama projenin başarısı için öncelikli rol oynamamaktadır.
  - İletişimin üst düzeyde tutulması gereksinimlerin toplanmasından başlayarak sürecin tüm adımlarında müşteri memnuniyetinin sağlanmasında önemlidir.
  - Yazılım ekibi ile yazılımı kullanacaklar arasında iyi bir iletişimin olması, sorunların erken fark edilmesini ve hızı kesilmeden projenin devam etmesini sağlar.

Uçdeğer Programlama - Prensipler (2)

- Geribildirim
  - Yazılım geliştirme süreçlerinde en önemli konulardan biri kalitedir. XP projelerinde kalite geribildirim üzerinden sağlanır.
  - Geribildirim, sorular sormak ve cevaplardan öğrenmek demektir. Müşterinin gerçekten ne istediğini anlamak için ona soru sormak gerekir. Ne kadar erken geri bildirim alınırsa, o kadar çabuk çözüm üretilir .
  - Geribildirim üç şekilde olur: Testlerden geribildirim, müşteriden geribildirim ve yazılım ekibinden geribildirim. Geribildirim sayesinde ortaya çıkabilecek yüksek maliyetli hatalar ortadan kalkabilir.
- Cesaret
  - Gerektiği zaman zor kararlar verebilmeyi savunur. Yazılımcı yeteri kadar cesur olmazsa hata yapma korkusu ile yazılımı müşteriye götürmekten çekinebilir; bu ise geribildirimin geç alınmasına, yapılan hatanın geç anlaşılmasına ve maliyetin artmasına neden olabilir.
  - Ayrıca yazılımcı yaptığı hatayı kabul etme cesaretini gösterebilmelidir.

Uçdeğer Programlama – Yaşam Döngüsü

- *Keşif Aşaması*
  - Müşteri ilk yayımda olmasını istediği kullanıcı hikayelerini ("user story") oluşturur.
  - Eş zamanlı olarak proje ekibi, teknik altyapı için proje kapsamında kullanacakları araçları, teknolojiyi ve pratikleri araştırıp öğrenmeye çalışır.
- *Planlama Aşaması*
  - Müşteri ve yazılım ekibi yineleme ve sürüm planlarını oluşturur. Yineleme planlaması için öncelikle yazılım ekibi, kullanıcı hikayelerini gerçekleştirme süreleri ile ilgili tahminde bulunur. Müşteri ise kullanıcı hikayelerine öncelik sırası vererek, yinelemelerde hangi kullanıcı hikayelerinin öncelikli olarak gerçekleştirileceğini belirler.
- *Yinelemeden Sürüm Üretilmesi*
  - Kullanıcı hikayelerinin gerçekleştirimi yapılır. Yazılım ve test ekibi tarafından testler gerçekleştirilir.
  - Her yineleme sonunda, o yineleme kapsamında gerçekleştirilen kullanıcı hikayelerinin işlevlerini kontrol etmek için müşteri tarafından kabul testleri yapılır. Her yineleme sonunda müşteriye çalışır bir yazılım sistemi sunulur. Bu şekilde müşterinin sistem hakkındaki görüşleri alınır.
- *Bakım Aşaması*
  - Yazılım sisteminin bakımının ve geliştirilmesinin yapıldığı aşamadır. Bu aşamada kullanıcılar için eğitim seminerleri hazırlanır ve yazılıma küçük çapta eklemeler ve sistem hatalarının giderilmesi için işlemler yapılır. Aynı zamanda sonraki yinelemelerin gerçekleştirimi de devam eder.