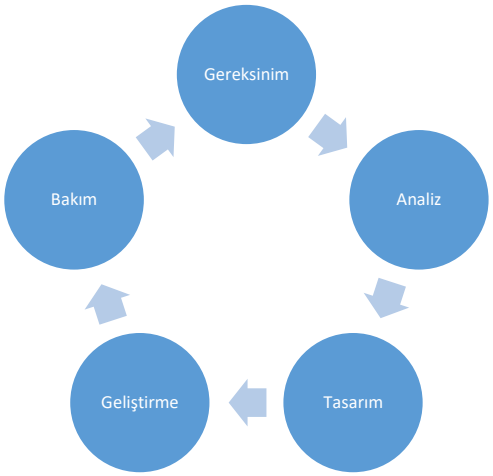
YAZILIM YAŞAM-DÖNGÜ MODELLERİ

Günümüzde bir yazılım projesinin geliştirilmesi sadece kodlama ve testten oluşmamaktadır. Bir yazılım projesinin geliştirilmesi sürecinde; *gereksinim, analiz, proje planlama, tasarım, kodlama, entegrasyon ve test aşamaları yer almaktadır.* Geliştirilecek bir yazılım projesinin planlamasından başlayarak müşteriye teslimatına kadar geçirmiş olduğu bütün aşamalara ve bu aşamaların oluşturduğu döngüye 'Yazılım geliştirme yaşam-döngüsü (software development life-cycle – SDLC)' adı verilir.



Kaynağa göre değişebilir ama anlatmak istenilen şey aynıdır. Hayat bir döngü içinde dönmektedir. Hepimiz hayatın içinde belli döngülerde yaşıyoruz. İnsanlar doğuyorlar, yaşıyorlar, ölüyorlar sonra toprak oluyor ve sonra o toprak başka hayatlara kaynak oluyor. Yağmur yağar, sular birikir, sonra buharlaşır, tekrar bulut olur ve tekrar yağmur yağar. Bir döngü her zaman vardır. Dolayısıyla yazılımın da kendi doğal yaşam döngüsü olduğunu ve bu döngünün doğru analiz edilmesi durumunda, doğru kontrollerin yapılması durumunda bizi başarıya götüreceğini unutmamalısınız.

TEORİDE YAZILIM GELİŞTİRME SÜRECİ

Teoride yazılım geliştirme süreci, bir işe sıfırdan başlayarak, gereksinim, analiz, tasarım, ve gerçekleştirme adımlarının doğrusal bir biçimde tanımlanması ile ifade edilir. **Ancak, pratikte yazılım geliştirme süreci iki durum göz önünde bulundurulduğunda oldukça farklıdır. Bu durumlardan ilki, yazılım geliştiriciler sonuçta insandır ve hata yapabilirler. İkinci durum ise, yazılım geliştirme sürecinde müşterinin istekleri değişebilir.** *Bu yüzden yazılım geliştirme süreci doğrusaldır demek doğru bir ifade olmaz.*

Yazılım geliştirme süreci, yazılım işlevleri ve ihtiyaçları sürekli değiştiği ve genişlediği için mutlaka bir döngü biçiminde düşünülmelidir. Döngü içerisinde herhangi bir aşamada geriye dönmek ve tekrar ilerlemek söz konusudur.

YAZILIM GELİŞTİRME YAŞAM DÖNGÜSÜ TEMEL ADIMLARI

Genelde beş temel adımdan oluşsa da farklı kaynaklarda adım sayısında değişiklik olabilir.

- Gereksinim (Requirements): Müşteri gereksinimlerinin elde edildiği ve fizibilite çalışmasının yapıldığı aşamadır.
- Analiz (Analysis): Sistem gereksinimlerinin ve işlevlerinin ayrıntılı olarak incelendiği aşamadır. Var olan işler incelenir, temel sorunlar ortaya çıkarılır.

- Tasarım (Design): Belirlenen gereksinimlere yanıt verecek yazılım sisteminin temel yapısının oluşturulduğu aşamadır. Mantıksal tasarım ile önerilen sistemin yapısı (modüller, akış diyagramı vb.), fiziksel tasarım ile de yazılımı içeren bileşenler ve bunların ayrıntılarını (veri yapıları, ekran tasarımları vb.) belirlenir.
- Gerçekleştirme (Implementation): Modüllerin kodlandığı, birleştirildiği, test edildiği ve kurulum çalışmaların yapıldığı aşamadır.
- Bakım (Maintenance): Yazılım tesliminden sonra hataların giderildiği ve yeni eklentilerin eklendiği aşamadır.

SDLC Yaklaşımının Avantajları;

- Bir sonraki aşamada problemin düzeltilmesi mümkün olabilir dolayısıyla risk düşürücü bazı özellikleri vardır.
- Gerçek dünyadaki problemlerin geliştirme süreçlerine dahil edilmesi söz konusudur.
- Bilginin, enformasyonun sistematik olarak analizi yapılabilir.
- Plan aşamasında, analiz aşamasında veya tasarım aşamasında bir problem çıktığında nerede çıktığı, yeni adımda eklenecek şeylerin hangi adımda eklenmesi gerektiği ortaya çıkarılabilir.
- Modüler yaklaşım uygulanabilir. Bir modüldeki problemin diğer modüllere yansıması engellenebilir. Bir modülün tamamen iptal edilebilmesi söz konusu olabilir.
- Adımların çıktıları ölçülür ve bir sonraki aşamaya geçmeden önce o adımın belli kalite standartlarının sağlaması şartı konulabilir.
- Dinamik ortam sürekli yaşayan, sürekli dönen bir ortamdır. Dolayısıyla dinamik ortamlar için uygundur. Örneğin; bir inşaat projesi yapılacak olsun ve genelde 100 yıl boyunca o inşaatın ayakta kalmasını beklenir. Bu çok dinamik bir proje değildir. Yazılım projeleri ise çok kısıtlı sürelerde çok büyük değişiklikler geçirir.

SDLC Yaklaşımının Dezavantajları;

- Uzmanlık gerektirir. Yönetim ile ilgili belli tecrübelere ihtiyaç vardır. En az bir uzmanın projeye dahil edilmesi hız kazandırır. Genelde, büyük sistemler için uygundur. Örneğin; ufak bir proje yazılacak. Hesap makinesi programı yazılacak. İyi bir programcı 2-3 saat içinde hesap makinesi programı yazabilir. Bu proje için SDLC uygulamak gerekli değildir. Hesap makinesi ile ilgili çok fazla bir şey istenmez. Bunun analizlerinin yapılması, tasarımlarının yapılması kalite standartlarının belirlenmesi mümkün. Fakat bu yaklaşım 3 saatlik projeyi 1-2 aya çıkabilir. 3 saatlik projenin maliyeti bir programcı için 100 lira civarıysa 1 aya çıkarttığında bu rakam 10 bin lira civarına çıkabilmektedir. *Genel anlamda belirtecek olursak büyük sistemler için daha uygun bir yaklaşımdır.*
- Bir diğer dezavantajı ise adım testleridir. Genelde her adım sonrasında test yapılması mümkündür ama bu yaklaşımda genelde adım testleri atlanır. Adımlardan sonra bir

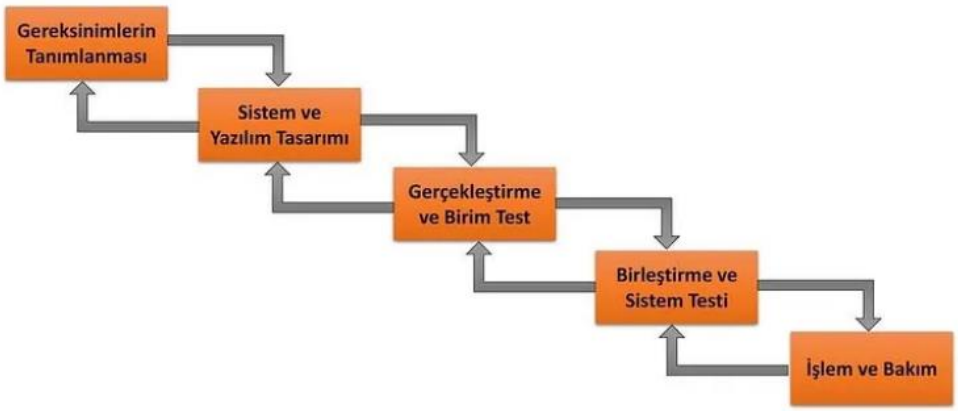
tanımlama yapıldı. Bu tanımlamalar ile ilgili ön/son koşullar oluşturup test yapılması, maket uygulamalar yapılması müşteri ile bunların tartışılması, başarı kriterlerinin belirlenmesi mümkündür. Fakat SDLC da bu adımları göremeyiz. Biraz hızı arttırmak açısından bu şekilde uygulanmaktadır. SDLC bir yaşam döngüsü olduğu için bir sonraki dönüşte adımları test etme özelliği avantaj sağlamaktadır. Bunların adım adım test edilmesi mümkündür. Hatanın devamlılığı engellenemez. Tasarımda çıkan bir hata ancak bir döngüde ele alınıp çözülebilir. Tasarımdaki hata diğer aşamalara da taşınmak zorundadır.

Günümüzde en yaygın kullanılan SDLC modelleri;

- Şelale Yaşam Döngü Modeli (Waterfall Life Cyle Model),
- V Süreç Modeli,
- Helezonik (Spiral) Model,
- Evrimsel Geliştirme Süreç Modeli,
- Artırımsal Geliştirme Süreç Modeli,
- Kodla ve Düzelt Yaşam-Döngü Modeli,
- Hızlı Prototipleme Yaşam-Döngü Modeli

ŞELALE YAŞAM DÖNGÜ MODELİ (WATERFALL LIFE CYLE MODEL)

En eski, en bilinen ve en temel yaşam döngü modelidir. Bu yaşam döngü modeli birçok hükümet tarafından kamuya özel yazılım geliştirme projelerinde standart olarak şartname dokümanlarında yer almıştır. Yakın zamana kadar en popüler yaşam döngü modellerinden biri olduğu bilinmektedir. **Gereksinimleri baştan tanımlanmış sistemlerin geliştirilmesinde daha çok tercih edilmektedir.** Bu yaşam döngü modelinde oluşturulacak sistemlerin her birini bir proje olarak ele almak gerekmektedir.



Şelale modelinde uygulanan adımlar şekilde gösterilmiştir.

- Şelale modelinde aşamaların en az birer kez tekrarlanması ile yazılım projesi geliştirilir. Bu modelde işler aşama aşama yapılır. Bir aşama bitmeden diğerine geçilmez. Her aşamanın sonunda dokümantasyon yazılmalıdır. Eğer bir aşamada dokümantasyon ve test olmamışsa o aşamanın tamamlandığı kabul edilmez.
- Şelale modelinde, proje içerisindeki dokümantasyon ayrı bir süreç olarak değil üretimin doğal bir parçası olarak ele alınır. Ayrıca, bu modelde aşamalar arasındaki geri dönüşlerin nasıl olacağı da tanımlıdır.
- Her ne kadar model içerisinde aşamalar arasında geri dönüşler yapılabilse de analiz aşamasında mümkün olan tüm detayın tasarıma yansıtılabilmesi için müşteri ve sistem gereksinimlerin en ince ayrıntısına kadar belirlenmesi gerekir. Bu yüzden tasarım aşamasında geliştirilecek yazılımın tüm gereksinimlerini karşılayacak şekilde detaylı bir çalışma gerektirmektedir. Bu nedenle belirsizlik içermeyen tam anlaşılmış ve değişme ihtimali olmayan projeler için daha uygundur.
- Bu yaklaşımın temel amacı, proje süresince değişikliklere izin vermeyerek, kapsam, zaman ve kaynaklar gibi faktörleri baştan sabitlemek ve büyük bir risk faktörü olan değişimi etkisiz kılmaktır.
- Şelale modeli disiplinli ve istikrarlı bir modeldir ve proje elemanlarına projenin iyi bir şekilde yönetildiği ve her şeyin kontrol altında olduğu hissiyatını verir.
- Değişiklik, geliştirme veya düzeltme maksatlı geriye dönüşler şelalede akıntıya karşı yüzmeye benzetilir, zor ve maliyetlidir. Bu yüzden değişikliklere cevap veremez ve süreç içerisinde gelen istekler çoğunlukla kabul edilmez.
- Herhangi bir safhada tespit edilmemiş bir olumsuzluk sonraki bütün safhaları etkiler.
- Yapısal özelliği ve sıkı kontrolleri nedeniyle esnek olmayan, yavaş ve hantal bir doğası vardır.

ŞELALE MODELİNDE DOKÜMANTASYON YAZIMI

Şelale modelinde yazılım yaşam döngüsünde **dokümantasyon** çok önemli bir yer tutar. Çünkü her aşama, bir önceki aşamanın çıktısına dayanır ve her aşama tamamlandığında bir doküman üretilir. Bu dokümanlar, proje sürecinin şeffaflığını ve izlenebilirliğini sağlar.

- **Gereksinim Dokümanı (Requirements Document)**

Örneğin;

Gereksinim 1: Kullanıcı, uygulama üzerinden hesap açabilmelidir.
Gereksinim 2: Uygulama, kullanıcı adı ve şifreyle kimlik doğrulaması yapmalıdır.
Gereksinim 3: Kullanıcı, bakiye sorgulama işlemi yapabilmelidir.
Gereksinim 4: Sistem, bir işlemde maksimum 3 saniye bekleme süresi sağlamalıdır.

- **Sistem Tasarımı Dokümanı (System Design Document)**

Örneğin;

Sistem Mimarisi: Uygulama istemci ve sunucu mimarisiyle çalışacaktır. İstemci, kullanıcı etkileşimini yönetirken, sunucu veritabanı işlemlerini gerçekleştirecektir.

Veritabanı Tasarımı: Kullanıcılar tablosu, işlemler tablosu ve bakiye tablosu olacak şekilde üç ana tabloya sahiptir.
Arayüz Tasarımı: Ana ekran, kullanıcı adı ve şifre girişi için iki metin kutusu içerecek, altında "Giriş" butonu olacak.

- **Kodlama ve Uygulama Dokümanı (Code Documentation):** Yazılımın kaynak kodlarını açıklamak, kodun nasıl çalıştığını anlamayı kolaylaştırmak amacıyla kullanılan dokümantasyondur. Kodun açıklamaları, sınıflar, fonksiyonlar, metodlar hakkında açıklamalar içerir.

Özellikleri:

Fonksiyon ve Metod Açıklamaları: Yazılımdaki her fonksiyonun ne yaptığına dair açıklamalar.

Değişken Açıklamaları: Kodun içindeki her değişkenin ne amaçla kullanıldığını anlatan açıklamalar.

Sınıf Açıklamaları: OOP (Object-Oriented Programming) kullanılıyorsa, her sınıfın ne işlevi olduğu açıklanır.

- **Test Planı Dokümanı (Test Plan Document)**

Örneğin;

Test Senaryosu 1: Kullanıcı doğru kullanıcı adı ve şifreyle giriş yapmalıdır.
Test Durumu: Kullanıcı adı: "cisem", Şifre: "123456"
Beklenen Sonuç: Giriş başarılı olmalı ve kullanıcı ana sayfaya yönlendirilmelidir.
Test Senaryosu 2: Yanlış kullanıcı adı ile giriş yapılmamalıdır.
Test Durumu: Kullanıcı adı: "deneme", Şifre: "password123"
Beklenen Sonuç: "Hatalı kullanıcı adı veya şifre" mesajı gösterilmeli, kullanıcı giriş yapmamalıdır.

- **Kullanıcı Kılavuzu (User Manual)**

Örneğin;

Kurulum Adımları:

Yazılımın kurulum dosyasını indiriniz.

Kurulum dosyasını çalıştırın ve ekrandaki adımları takip edin.

Kurulum tamamlandıktan sonra, yazılım otomatik olarak başlatılacaktır.

Ana Ekran:

Kullanıcı Adı Alanı: Kullanıcı adı girebileceğiniz bir metin kutusu.

Şifre Alanı: Şifrenizi girebileceğiniz bir metin kutusu.

Giriş Butonu: Kullanıcı adı ve şifrenizi girerek giriş yapmak için tıklamanız gereken buton.

- **Bakım ve Destek Dokümanı (Maintenance and Support Document)**

Örneğin;

Hata Çözümleme:

Hata: Kullanıcı girişi sırasında "Hatalı şifre" mesajı alınıyor.

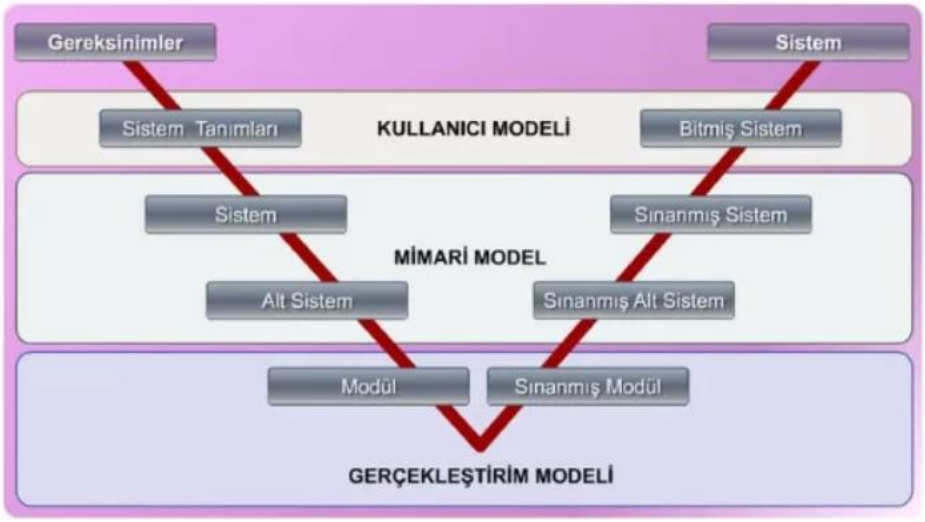
Çözüm: Kullanıcı adı ve şifreyi kontrol edin, şifre sıfırlama seçeneğini deneyin.

Yazılım Güncellemeleri:

Yeni güncellemeleri indirin.

Güncelleme dosyasını çalıştırın ve talimatları izleyin.

V SÜREÇ MODELİ NEDİR?



- V süreç modeli adından da anlaşılacağı gibi 'V' harfi yapısında bir yol izlenir ve adımlar bu şekilde gerçekleştirilir.
- Şekilde görüldüğü gibi modelin sol tarafı üretimi, sağ tarafı ise sınama (test) işlemini ifade eder.
- Şelale modelinin gelişmiş halidir. Bu süreç modeli belirsizliklerin az, iş tanımlarının belirgin olduğu bilgi teknolojileri (BT) projeleri için uygun bir modeldir. Kullanıcının projeye katkısı bulunmaktadır. Kullanıcı yazılım projesinin geliştirilmesi sürecinde projenin içerisindedir. Aynı zamanda adımlar doğrusal ilerlemek yerine kodlama evresinden sonra yukarı yönde kıvrılarak V şeklini oluşturur.

V Süreç modelinin 3 temel çıktısı vardır. Bunlar;

- Kullanıcı Modeli: Geliştirme sürecinin kullanıcı ile olan ilişkilerini tanımlamakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.

- Mimari Model: Sistem tasarımı ve alt sistem ile bütün sistemin test işlemleri yapılmaktadır.
- Gerçekleştirim Modeli: Yazılım modüllerinin kodlanması ve testine ilişkin fonksiyonlar ele alınmaktadır.

Senaryo: Kamu Binaları için Akıllı Güvenlik Sistemi

Proje, bir kamu binasında güvenliği artırmak amacıyla bir yazılım geliştirmeyi hedefliyor. Yazılım, güvenlik kameraları, kapı erişim kontrolleri, alarm sistemleri ve acil durum müdahale protokollerini yöneten bir uygulama olacak. Bu projenin V süreç modeline göre düzenlemelerini gerçekleştiriniz?

Gereksinim Analizi (Requirements Analysis)

- **Geliştirme Aşaması:**
  İlk aşamada, proje paydaşları (güvenlik uzmanları, bina yöneticileri, kullanıcılar) ile toplantılar yapılarak yazılımın işlevsel gereksinimleri belirlenir. Kullanıcıların sistemden beklentileri ve yazılımın temel özellikleri (kameraların yönetimi, erişim kontrolleri, alarm bildirimleri, vb.) tanımlanır.

Örnek Gereksinim:

"Sistem, binada her katta en az 2 güvenlik kamerasını izleyebilmeli."

"Kullanıcı, şüpheli bir hareket algılandığında anında alarm almalı. "

- **Test Aşaması:**
  Kabul Testi Planı (Acceptance Test Plan) hazırlanır. Yazılımın gereksinimlere uygunluğunu belirlemek için kullanılacaktır. Her bir gereksinim için başarı kriterleri belirlenir.

Sistem Tasarımı (System Design)

- **Geliştirme Aşaması:**
  Bu aşamada yazılımın genel mimarisi tasarlanır. Sistemin hangi bileşenlerden oluşacağı, bunların nasıl etkileşimde bulunacağı belirlenir. Örneğin, sistemde bir **veritabanı** olacak, bir **kullanıcı arayüzü** olacak ve **güvenlik kameraları** ile **alarm sistemleri** entegre edilecektir.
- **Test Aşaması:**
  Sistem Testi Planı (System Test Plan) hazırlanır. Tüm bileşenlerin uyumlu bir şekilde çalışıp çalışmadığını test etmek için kullanılan planı içerir. Örneğin, bir güvenlik kamerasının anlık görüntüleri doğru şekilde veritabanına kaydediliyor mu, alarm sistemine doğru şekilde veri iletimi yapılıyor mu gibi testler için bir plan yapılır.

Modül Tasarımı (Module Design)

- **Geliştirme Aşaması:**
  Yazılımın her bir modülünün tasarımı yapılır. Bu aşama, her bir bileşenin iç tasarımını,

algoritmalarını ve kullanıcı arayüzü unsurlarını içerir. Örneğin, "Kamera Modülü", "Alarm Yönetim Modülü" ve "Kullanıcı Yönetim Modülü" gibi alt modüller tasarlanır.

- **Test Aşaması:**
  Birim Testi Planı (Unit Test Plan) hazırlanır. Bu testler, her modülün bağımsız olarak doğru çalışıp çalışmadığını kontrol etmek için kullanılır. Örneğin, "Kamera Modülü" için, sistemin doğru bir şekilde video akışı sağladığı, görüntülerin kaydedildiği ve görüntü kalitesinin belirli bir düzeyde olduğu test edilir.

Kodlama (Coding)

- **Geliştirme Aşaması:**
  Yazılımın kodlanma aşamasıdır. Her bir modülün geliştirilmesi ve fonksiyonların yazılması bu aşamada yapılır. Örneğin, "Kamera Modülü"nün kamera verisini alması, görüntüyü işleyip ekranda göstermesi ve kaydetmesi için gerekli kod yazılır.

- **Test Aşaması:**
  Kodlama tamamlandıkça, **birim testleri** yapılır. Her modül bağımsız olarak test edilir. Örneğin, kamera verisi işlenip kaydedildikten sonra, görüntünün düzgün kaydedilip kaydedilmediği test edilir.

Entegrasyon Testi (Integration Testing)

- **Geliştirme Aşaması:**
  Bu aşamada, tüm modüller birleştirilir ve entegrasyon işlemi yapılır. Modüllerin birbirleriyle uyumlu çalışıp çalışmadığı test edilir. Örneğin, **Kamera Modülü** ile **Alarm Yönetim Modülü** entegre edilir ve kameradan gelen herhangi bir şüpheli hareket algılandığında alarm modülü doğru bir şekilde uyarı veriyor mu kontrol edilir.

- **Test Aşaması:**
  Entegrasyon Testi yapılır. Modüllerin birlikte çalışıp çalışmadığı, her bir bileşenin doğru şekilde entegre olup olmadığına dair testler yapılır.

Sistem Testi (System Testing)

- **Geliştirme Aşaması:**
  Tüm sistem birleştirilir ve yazılımın tüm özelliklerinin bir arada çalışıp çalışmadığına bakılır. Kullanıcıların kameraları izleyip alarm durumlarını kontrol etmesi, sistemin kullanıcı dostu arayüzünü kullanarak kolayca işlem yapabilmesi sağlanır.

- **Test Aşaması:**
  Sistem Testi yapılır. Burada yazılımın tamamı bir arada test edilir. Sistem gereksinimlerinin tümü test edilir. Ayrıca güvenlik, performans ve yük testleri gibi kapsamlı testler de yapılır.

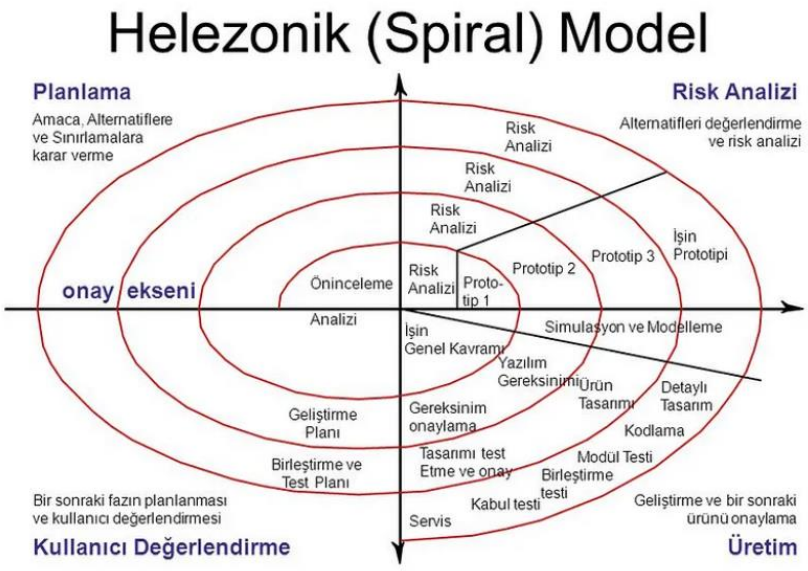
Bakım ve Destek (Maintenance and Support)

- **Geliştirme Aşaması:**
  Yazılımın üretime alındıktan sonra, bakım ve destek süreci başlar. Burada yazılımda oluşan hatalar düzeltilir, yeni özellikler eklenir ve sistem güncellemeleri yapılır.

- **Test Aşaması:**
  Regresyon Testleri yapılır. Yeni eklenen özelliklerin mevcut işlevsellikleri bozup bozmadığını kontrol etmek için, yazılımın daha önce test edilmiş bölümleri yeniden test edilir. Ayrıca yazılımın güvenliği ve performansı izlenir.

V Süreç Modeli, yazılım geliştirme sürecinde her adımın, eş zamanlı bir test aşamasıyla ilişkilendirildiği, bu sayede yazılımın kalitesini sürekli artıran bir yaklaşımdır. Testlerin sürekli yapılması, yazılımın hatalarını erken aşamalarda tespit etmeye ve çözmeye yardımcı olur. Ancak, gereksinimlerin başlangıçta net olması ve değişikliklere karşı esneklik olmaması, bu modelin zorluklarını oluşturur.

HELEZONİK (SPİRAL) MODEL NEDİR?



1998'de Barry Boehm tarafından risk odaklı bir yazılım geliştirme süreci olarak önerilmiştir. Bu modelde yazılım geliştirme süreci, bir aktiviteden diğerine birtakım geri dönüşleri takip etmek yerine spiral olarak temsil edilir.

Bu süreç modeli, olası değişikliklerin proje risklerinden biri olduğunu varsayar. Bu riskleri azaltmak içinde risk yönetim faaliyetlerini içerir. Spiralde 4 döngü bulunmaktadır. Bunlar;

- **Planlama:** Geliştirilecek prototip için plan yapılmasını, amaçların belirlenmesini ve bir önceki adımda üretilen prototipe entegrasyonu sağlar.
- **Risk Analizi:** Risk seçeneklerinin araştırılmasını ve risklerin belirlenmesini sağlar.
- **Üretim:** Prototiplerin geliştirilmesini içerir. Her aşamanın sonunda bir prototip oluşturulur ve bu prototipler geliştirilerek kullanıma hazır son prototip ortaya çıkarılmış olunur.
- **Kullanıcı Değerlendirmesi:** Prototip ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeleri ele alır.

- Helezonik modeli kullanmanın en önemli avantajı, üretim süresi boyunca prototiplerin geliştirilmesi ve geliştirilen prototiplerin kullanıcı tarafından sınanmasıdır. Aynı zamanda gerek proje sahibinin gerekse yöneticilerin, proje boyunca çalışan prototip yazılımlarla karşılaşmaları projeyi daha kolay takip edebilmelerini ve hak ediş planlaması yapabilmelerini sağlar.

SENARYO: Sağlık Takip Uygulaması

Bu projede, kullanıcıların günlük aktivitelerini, uyku düzenlerini, beslenme alışkanlıklarını ve genel sağlık durumlarını takip edebileceği bir mobil uygulama geliştirilmesi hedefleniyor. Uygulama, başlangıçta temel özelliklere sahip olacak, ancak her yinelemede yeni işlevler eklenecek ve uygulama geliştirilip iyileştirilecektir. Bu projenin Helezonik (spiral) modeline göre düzenlemelerini gerçekleştiriniz?

Planlama ve Gereksinim Toplama (Planning and Requirements Gathering)

- **İlk Yineleme:**
  Proje başlangıcında, kullanıcılar ve paydaşlarla toplantılar yapılarak, uygulamanın genel gereksinimleri belirlenir. Proje için belirli hedefler, kullanılacak platformlar ve temel işlevler tanımlanır. Örneğin, başlangıçta kullanıcıların günlük adım sayısını izleyebilmesi, uyku düzenini takip etmesi ve sağlık geçmişine dair bazı temel verileri saklayabilmesi gibi işlevler ele alınır.

- **Risk Analizi:**
  İlk yineleme aşamasında, yazılımın teknik, fonksiyonel ve operasyonel riskleri belirlenir. Örneğin, uygulamanın veri toplama ve senkronizasyon sürecinde yaşanabilecek problemler, kullanıcı verilerinin gizliliği gibi konular ele alınır. Bu riskler değerlendirilir ve önleyici önlemler alınır.

Tasarım ve Prototip Oluşturma (Design and Prototyping)

- **İkinci Yineleme:**
  İlk yinelemeden elde edilen gereksinimlere dayanarak uygulamanın temel tasarımı yapılır. **Kullanıcı arayüzü (UI)** tasarımı, temel fonksiyonel özellikler için taslaklar ve bir prototip oluşturulur.

Örnek Tasarım:

Uygulama, kullanıcı adım sayısını ana ekranda gösteren bir panel içerecek.

Kullanıcılar uyku verilerini bir takvim aracılığıyla görebilecekler.

Ana ekran, kullanıcıyı sağlık verilerini eklemeye teşvik edecek basit ve kullanıcı dostu olacak.

- **Risk Analizi ve Değerlendirme:**
  Prototipin oluşturulması sırasında teknik riskler ve kullanıcı deneyimi ile ilgili olası zorluklar ele alınır. Örneğin, uygulamanın farklı mobil cihazlarda uyumlu çalışıp çalışmadığı test edilir. Ayrıca, kullanıcıların bu basit prototipe nasıl tepki verdikleri değerlendirilir.

Uygulama Geliştirme ve Test (Development and Testing)

- **Üçüncü Yineleme**
  Uygulamanın ilk sürümü kodlanmaya başlanır. Bu aşamada temel işlevler, veritabanı bağlantıları, adım sayma sensörlerinin entegrasyonu ve veri kaydı gibi özellikler geliştirilir.

Örnek Geliştirme:

Kullanıcı giriş ekranı ve profil oluşturma özellikleri geliştirilir.

Adım sayma özelliği, telefonun sensörlerine entegre edilir.

Uygulama, günlük adım sayısını kaydedip kullanıcının profilinde saklar.

- **Test**
  Yazılımın her aşamasında, özellikle geliştirme aşamalarında **birim testleri** ve **entegrasyon testleri** yapılır. Uygulamanın adım sayma fonksiyonu ve kullanıcı verilerinin doğru şekilde kaydedildiğinden emin olunur. Ayrıca, güvenlik testleri yapılır.

- **Risk Analizi ve Değerlendirme**
  Testlerin sonunda, yazılımın işlevsel olmayan gereksinimlerini (örneğin, performans ve güvenlik) ele alacak yeni risk analizleri yapılır. Eğer uygulama, belirli cihazlarda düzgün çalışmıyorsa, buna yönelik çözümler geliştirilir.

Geri Bildirim ve İyileştirme (Feedback and Refinement)

- **Dördüncü Yineleme:**
  İlk kullanıcı testlerinden elde edilen geri bildirimler doğrultusunda, uygulamanın kullanıcı arayüzü ve işlevselliği iyileştirilir. Örneğin, kullanıcılar daha fazla sağlık verisi eklemek isteyebilir, veya adım sayma işleminin doğruluğunu sorgulayabilir.

Örnek İyileştirme:

Uyku kalitesini kaydetmek için ek parametreler (örneğin, uyandığı anlar) eklenir.

Kullanıcı profili sekmesinde, kullanıcının günlük aktivite hedefleri görünür hale gelir.

- **Risk Analizi ve Değerlendirme:**
  Kullanıcılardan alınan geri bildirimlere göre yeni riskler ortaya çıkabilir. Örneğin, veri gizliliği konusunda ek önlemler almak gerekebilir veya sensörlerin doğruluğunun daha iyi olması için geliştirmeler yapılabilir. Bu riskler değerlendirilir ve uygulamanın daha sağlam bir versiyonu için planlar yapılır.

Sürüm ve Yineleme (Release and Iteration)

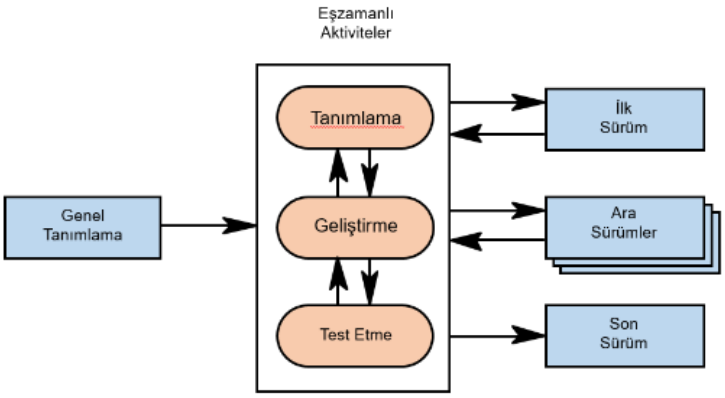
- **Beşinci Yineleme ve Sürüm:**
  Bu aşamada, uygulamanın daha stabil ve kullanıcı dostu bir sürümü oluşturulur. Kullanıcı geri bildirimleri alındıktan sonra, tüm fonksiyonel özellikler tamamlanır ve

uygulama son kullanıcıya sunulur. Uygulama, tüm platformlarda (Android ve iOS gibi) çalışacak şekilde yayına alınır.

- **Sonraki Yinelemeler:**
  Uygulama piyasaya sürüldükten sonra, kullanıcı verileri toplandıktan sonra ek özellikler ve iyileştirmeler yapılır.

Helezon Modeli, yazılım geliştirme sürecinde sürekli risk analizi yaparak iteratif bir şekilde ilerleyen bir modeldir. Bu model, özellikle belirsizliğin ve risklerin fazla olduğu projelerde çok faydalıdır. Yazılım, her yinelemede test edilip geliştirilerek olgunlaşır ve bu süreçte kullanıcı geri bildirimleri önemli bir rol oynar. Bu şekilde, projede karşılaşılan riskler ve gereksinim değişiklikleri en aza indirgenir.

EVRİMSEL GELİŞTİRME SÜREÇ MODELİ NEDİR?



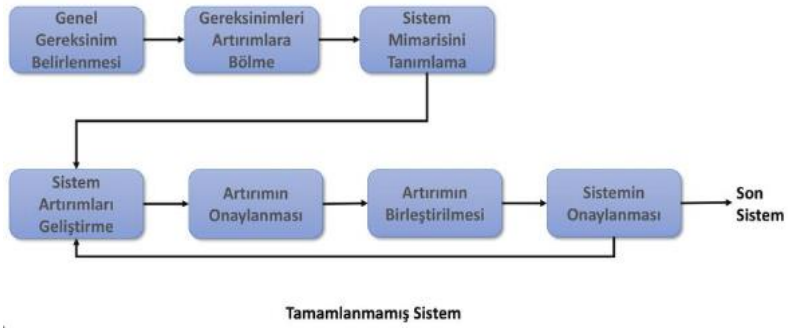
Evrimsel geliştirme süreç modeli ilk tam ölçekli modeldir. Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlarda kullanılmak üzere geliştirilecek projeler için önerilir.

Bu süreç modelinde sistem, zaman içinde kazanılan anlayışa göre sürümler şeklinde geliştirilir. Müşteriden elde edilen gereksinimlerle projenin geliştirilmesine başlanır ve ilk sürüm üretilir. Müşteri tarafından gelen taleplere göre sisteme yeni özellikler eklenerek yeni sürümler elde edilir. Modelin başarısı geçirdiği ilk evrimin başarısına bağlıdır.

Yazılım projelerinde bu süreç modelini kullanmanın en önemli avantajı; müşteri gereksinimlerinin anlaşılmasını kolaylaştırır. Bunun yanında projenin geliştirilmesi süresince ortaya çıkabilecek riskleri ve hataları azaltır.

Bu modelin en önemli dezavantajı, konfigürasyon ve değişiklik yönetiminin zor olmasıdır. Ayrıca, sistemin bakımını da yapmak zordur.

ARTIRIMSAL GELİŞTİRME SÜREÇ MODELİ NEDİR?



Artırımsal süreç modeli, şelale modeli ile evrimsel geliştirme süreç modelini temel almaktadır. Bir yazılım ürünü geliştirilirken sürekli olarak değişen müşteri gereksinimlerini karşılayan ve ortaya çıkan kaçınılmaz hataların zamanında düzeltilmesini sağlayan bir süreç modelidir.

- Bu modelde sistemi tek seferde teslim etmek yerine, geliştirme ve teslim parçalara bölünür.
- Her teslim beklenen işlevselliğin bir parçasını karşılar.
- Müşteri gereksinimleri önemine göre önceliklendirilir ve öncelikli gereksinimler erken teslimlere dahil edilir.
- Müşteri gereksinimleri önemlerine ve birbirine bağımlıklarına göre sıralanarak her yinelemede bu gereksinimlerin bir kısmı tamamlanır.
- Bir parçanın geliştirilmesi başladığında, gereksinimler dondurulur. Olası değişiklikler sonraki teslimlerde ele alınır.
- Üretilen her teslim sürümü birbirini kapsayacak ve giderek artan sayıda işlev içerecek şekilde geliştirilir.

SENARYO: Web Uygulaması için Güvenlik İyileştirmeleri

Bir e-ticaret şirketi, kullanıcı bilgilerini güvenli bir şekilde işleyen bir web uygulaması geliştirmeyi planlıyor. Uygulama, artan güvenlik tehditlerine karşı daha sağlam hale getirilmek üzere artırımlı geliştirme modelini kullanarak her sürümde yeni güvenlik önlemleri ekleyecek şekilde düzenleyiniz?

İlk Artırım: Temel Şifreleme ve Güvenli Kullanıcı Girişi

Gereksinimler:

Kullanıcıların şifreleri şifrelenmiş olarak veritabanına kaydedilecek.

Giriş işlemi için temel güvenlik önlemleri alınacak (örneğin, HTTPS üzerinden şifreli bağlantılar).

Geliştirme:

İlk artırımda, kullanıcı şifreleri veritabanına güçlü bir şifreleme algoritmasıyla şifreli şekilde kaydedilir. Ayrıca, kullanıcıların giriş yapabilmesi için yalnızca HTTPS üzerinden bağlantıya izin verilir.

Test ve Geri Bildirim:

Şifreleme algoritmasının doğru çalışıp çalışmadığı ve HTTPS bağlantısının etkin olup olmadığı test edilir.

Kullanıcılar, şifre sıfırlama ve giriş işlemlerini test eder.

İkinci Artırım: Çift Faktörlü Kimlik Doğrulama (2FA)

Gereksinimler:

Kullanıcılar, giriş yaptıktan sonra Çift Faktörlü Kimlik Doğrulama (2FA) ile ek bir güvenlik adımı geçmelidir.

Geliştirme:

İkinci artırımda, kullanıcıların giriş yaptıktan sonra SMS veya mobil uygulama üzerinden bir doğrulama kodu girerek giriş yapmalarını sağlayan 2FA özelliği eklenir.

Test ve Geri Bildirim:

2FA sürecinin düzgün çalışıp çalışmadığı test edilir.

Kullanıcılar, SMS ile gelen doğrulama kodlarının düzgün şekilde alınıp alınmadığını test eder.

Üçüncü Artırım: Güvenlik Duvarı ve Otomatik Saldırı Tespiti

Gereksinimler:

Web uygulaması güvenlik duvarı (WAF) ile korunacak.

Otomatik saldırı tespiti yapılacak; SQL enjeksiyonu ve brute-force saldırıları gibi tehditler tespit edilip engellenecek.

Geliştirme:

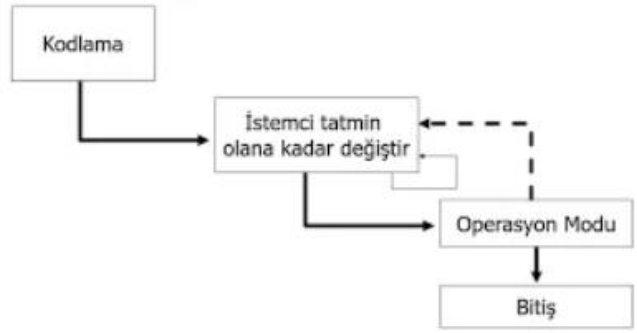
Bu artırımda, uygulamanın dış tehditlerden korunması için WAF (Web Application Firewall) entegre edilir. Ayrıca, sistemin brute-force saldırılarına karşı korunması için giriş denemeleri sınırlanır ve SQL enjeksiyon gibi saldırılara karşı önlemler eklenir.

Test ve Geri Bildirim:

Saldırı tespiti sisteminin doğru çalışıp çalışmadığı test edilir.

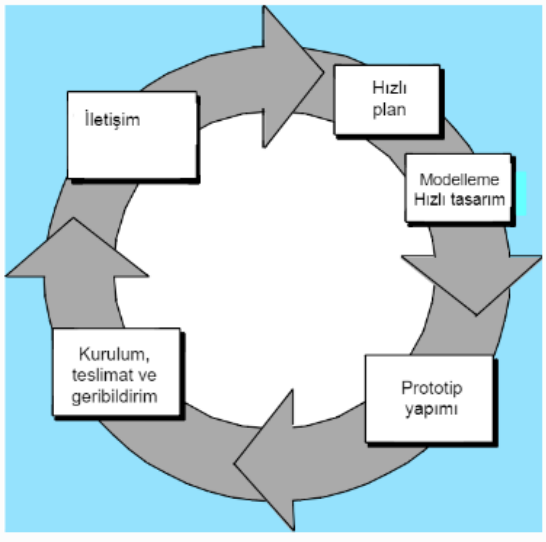
Güvenlik duvarının doğru şekilde engelleme yapıp yapmadığı kontrol edilir.

KODLA VE DÜZELT YAŞAM DÖNGÜ MODELİ NEDİR?



- Günümüzde birçok yazılım firmasının yazılım geliştirirken tercih ettiği modellerden biridir.
- Yazılım ürünü gereksinim, analiz ve tasarım safhalarının herhangi biri dikkate alınmaksızın direkt olarak kodlama ve test işlemine tabi tutulur.
- Özetle, ilk safhada yazılım ürününün ilk sürümü ortaya çıkarılır. Yazılım ürünü en son istenilen şekle gelinceye kadar devamlı geliştirilir.
- Bu yazılım geliştirme yaşam-döngü modeli yazılım geliştirmenin en kolay ancak en pahalı yoludur.
- Kodlama safhasından sonra bir yazılım ürünü içerisindeki değişikliklerin maliyeti çok yüksektir. Buna ilave olarak dokümantasyonun olmadığı için ürünün bakımını yapmak çok zordur.
- Takım çalışmasına uygun bir model değildir. Bireysel geliştiricilere uygundur.

HIZLI PROTOTİPLEME YAŞAM DÖNGÜ MODELİ NEDİR?



- Prototip model, bir yazılım sisteminin ilk sürümüdür.
- Bir problem ve olası çözümleri hakkında daha fazla bilgi edinmek, tasarım seçeneklerini denemek ve temel kavramları göstermek için prototip model kullanılır.
- Burada amaç, kullanıcının gereksinimlerini ortaya çıkarmaktır.
- Hızlı prototipleme yaklaşımı, yazılım maliyetlerinin kontrol altında tutulması için gereklidir.
- Ayrıca sistem paydaşlarının yazılım sürecinin erken aşamalarında prototip model ile yazılım ürününü tecrübe edebilmelerini sağlar. Bunun yanında bir yazılım prototipi, gerekli olabilecek değişikliklerin tahmin edilmesine yardımcı olmak üzere yazılım geliştirme süreci içerisinde kullanılabilir.
- Bu model şelale modeli benzerlik gösterir fakat geriye dönüşleri yoktur, yani doğrusal bir modeldir.

KAYNAKLAR

Bilişimde Sistem Analizi ve Tasarımı, Oya H. YÜREGİR, Nobel Kitabevi , ISBN: 975-85-61-05-7

Sommerville, I. (2011). Software engineering (ed.). *America: Pearson Education Inc*.

Yücalar, F. ve Borandağ, E. (2022). Yazılım Mühendisliğinde Modern Yaklaşımlar