**YAZILIM YAŞAM DÖGÜSÜ MODELLERİ**

**YAZILIM YAŞAM DÖNGÜSÜ NEDİR**

Aslında yazılım bir üründür ve yazılımın bir yaşam süreci vardır. Bu yaşam süreci birkaç aşamadan oluşur. Bu aşamalar sırasıyla planlama, analiz, tasarım, gerçekleştirme ve bakımdan oluşur. Yazılım üretildiği aşamadan itibaren işlevsellik ve gereksinimleri sürekli değişecek ve bu değişikliklerde yazılımın genişlemesine neden olacaktır. Dolayısıyla bu değişimler bir döngü olarak ele alınmak zorundadır. Bu döngü doğrusal veya tek yöne ilerleyen bir döngü değildir çünkü herhangi bir noktada geriye dönmek, geliştirme yapmak ve tekrar ilerlemek mümkündür.

**Planlama**

Belirlenen hedeflere ulaşmak için izlenecek yöntem, yapılacaklar listesi, görev dağılımı, kullanılacak kaynaklar, süre takvimi gibi olayların belirlenmesidir. Bir hedef belirlenir ve planlama süreci başlamış olur. Üretilecek olan ürün için müşteriden müşterinin gereksinimleri alınır ve yapılacaklar listesi ortaya çıkar. Projenin hedef kitlesi olan kullanıcılar belirlenir. Projedeki asıl amaç müşterinin istediği sürede projeyi tamamlamaktır.

**Analiz**

Analiz ihtiyaç ve beklentilerin ayrıntılı bir şekilde tanımlanmasıdır. Buradaki temel amaç yazılım mühendisi açısından mevcut yapılan plandaki çözümün incelenip müşteri isteğinin doğru bir şekilde anlaşılıp anlaşılmadığının ortaya çıkarılmasıdır. Bu aşamada gereksinimler detaylı bir şekilde incelenir ve projede nelerin istenildiği ile ilgili analiz çalışmaları da yapılır.

**Tasarım**

Tasarım, planlama aşamasında istenen, analiz sonucunda belirlenmiş gereksinimleri yerine getirecek ve geliştirilecek yazılımın üst seviye modelinin oluşturulması işlemidir.

**Gerçekleştirme**

Müşteriye teslim edilecek projenin kodlama aşamasıdır. Kodlama işlemi planlama veya çözümleme aşamasında belirtilen programlama dili, geliştirme ortamı ve teknolojilerin yardımıyla yapılır.

**Bakım**

Bakım süreci, yazılımın kullanılmaya başlandıktan sonra sistem üzerine değişiklik yapılması sürecidir. Test etme ve kurulum aşamasından sonra gerçek ortama kurulan yazılımın, hata giderme, güncelleme gibi işlemleri yapılır.

**Yazılım Yaşam Döngüsünün Avantajları ve Dezavantajları**

**Avantajları**

Yazılım yaşam döngüsünde her bir aşamadan sonra gözden geçirme imkânı olduğu için proje, projeyi geliştiren yazılımcılar ve müşteriler kontrolünde gelişmiş olur. Bu sayede hedeflere ulaşılıp ulaşılmadığı kontrol edilmiş olur. Gözden geçirmeler sayesinde hatlarda düzeltilmiş olur. Bu yaklaşımda sistemin planlanmasından başlayarak analiz, tasarım, kodlama gibi her bir aşama için detaylı dokümantasyon sağlanmış olur. Bu aşamalar sayesinde kullanıcının isteklerinin karşılanıp karşılanmadığı kontrol edilir.

**Dezavantajları**

Yazılım yaşam döngüsünün dezavantajlarından biri küçük ve sade projelerde maliyeti ve karmaşıklığı artırması bu yüzden küçük projeler için uygun değildir. Proje başladıktan sonra her şeyin ayrıntılı olarak bilinmesi mümkün olmadığı için istenildiği gibi bir proje ortaya çıkmayabilir. Sisteme kullanıcıların bütün taleplerinin eklenmesi durumunda içinden çıkılamayacak kadar karmaşık bir hal alabilir. Bunun sonucu olarak proje hedeflendiği gibi sonuçlanmayabilir. Her aşamada dokümantasyon olması ve dokümantasyonların fazla olması zaman kaybına yol açabilir.

**YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ**

Birçok yazılım yaşam döngüsü modeli var. Her modelin avantajları olduğu gibi dezavantajları da vardır. Ama bir modelin bir proje için en uygun model olduğu hakkında bir kural yok.

**Şelale(waterfall) Modeli**

Şelale modeli 1970 yılında Royce tarafından sistem mühendisliği sürecinden türetilen yazılım geliştirme sürecinin ilk yayınlanmış modelidir. Üretim sürecindeki temel adımları baştan sona en az bir kez izleyerek gerçekleşmektedir. Gereksinimleri iyi tanımlanmış ve kısa vadede üretilebilecek olan projeler için oldukça uygun bir modeldir. Bu model her aşamada yazılım gereksinimleri, tasarım belgeleri, kod ve benzeri olaylar ile aşama aşama ilerlemektedir. Her aşama başlamadan önce gerçekleştirmesi gereken faaliyetler vardır. Eğer ürün gereksinimleri iyi tanımlanmamış veya üretim süresi uzun sürecekse model işe yaramaz. Yazılım projeleri çok esnektir ve çok sık bir şekilde değiştiğinden şelale modeli yazılım projeleri için kötü yaklaşımlardan biridir denebilir. Bu modelin kilit noktası bir etkinlik tamamlandıktan sonra geri dönmemektir. Dolayısıyla her etkinliği ve aşamayı bir kez gözden geçirmek zorunludur. Kısaca bu model yazılım geliştirme sürecinin kullanıcı ihtiyaçlarını koda dönüştüren adım adım bir süreç olarak planlanabileceği varsayılır.

**Şelale Modelinin Avantajları ve Dezavantajları**

**Avantajları**

Projenin daha doğru bir şekilde ilerlemesini ve olabilecek hataların daha erken fark edilmesini sağlar. Proje daha rahat yönetildiği için maliyet aşımı olmadan ve zamanında teslim edilir. Her aşmadan sonra inceleme yapılması kullanıcı katılımını sağlar. Teknik açıdan zayıf ve deneyimsiz personel için iyi çalışır.

**Dezavantajları**

Müşteri ne istediğini eksiksiz olarak ifade etmelidir. Planlama ve belgeleme için çok fazla zaman harcanmaktadır. Gereksinimlerdeki değişiklikler ve hataları düzeltmek zor ve masraflıdır. Çoğu ihtiyacı önceden tahmin etmek zordur.

**V Süreç Modeli**

V modeli, aşamaların ardışık bir şekilde ilerlediği bir yazılım yaşam döngüsü modelidir. Bu modelde v şeklinde bir yol izlenir. V modeli şelale modelinin gelişmiş hali olduğu söylenebilir.



Bu döngü üç modelden oluşur: Kullanıcı Modeli, Mimari Modeli, Gerçekleştirim Modeli. Kullanıcı modelinde kullanıcının istekleri ve proje için tanımları kullanılır ve tamamlanmış hali teslim edilir. Mimari modelde projenin tasarımı ve bunların deneme işlemlerinden oluşur. Gerçekleştirim modeli ise kodlama ve bunların denenmesinden oluşur. Bu model belirsizliklerin az olduğu, iş tanımlarının ise belirgin olduğu projelerde kullanılır. Bu tür projelere örnek olarak Bilgi Teknolojileri için geliştirilen projeler verilebilir.

**V Süreç Modelinin Avantajları ve Dezavantajları**

**Avantajları** Oldukça disiplinli bir modeldir ve aşamaları birer birer tamamlanmıştır. Basit ve kolay anlaşılır bir modeldir. Her aşamadaki modelin sağlamlığı nedeniyle yönetilmesi kolaydır. Gerekliliği iyi anlaşıldığı küçük projeler için çok kullanışlıdır.

**Dezavantajları** Karmaşık ve nesne yönelimli projeler için kullanışlı değildir. Bir uygulama test aşamasındayken geriye dönüp değişiklik yapmak zordur. Uzun ve devam eden projeler için zayıf bir modeldir. İhtiyaçların yüksek risk altında olduğu projeler için uygun değildir. Aşamalarda tekrar bulunmaz ve risk çözümleme için ayrılan bir yer yoktur.

**Spiral (Helezonik) Model**

Spiral modeli ilk olarak 1988 yılında Barry Boehm tarafından tanımlanmıştır. Spiral modeli diğerlerinden ayıran özellik ise risk analizinin ön planda olması ve prototip oluşturulmasıdır. Risk analizi ön planda olduğu için hataları erken giderme imkânı tanıyabilir. Her bir aşama bir tasarım hedefi ille başlar, standart yazılım süreçlerinin bazı aşamalarını geçip müşteri yorumu ve görüşü ile biter. Her bir aşamada amaç iş değerini yakalamak ve bir önceki aşamanın sonuçlarından faydalanarak gelişen bir yapıda devam etmektir. 4 temel aşamadan oluşur: Planlama, her aşamada olan ara ürün için bir planlama yapılır. Risk analizi, risklerin araştırılması, belirlenmesi ve çözülmesi. Üretim, ara ürünün üretilmesi. Kullanıcı değerlendirmesi, oluşturulan ara ürünün sonucunda kullanıcıdan alınan geri dönütlerin değerlendirilerek diğer aşamaya geçilmesi.

**Spiral Modelin Avantajları ve Dezavantajları**

**Avantajları**

Gereksinimler daha doğru bir şekilde belirlenebilir. Kullanıcıların sistemi başta görebilmesini sağlar. Geliştirme küçük parçalara bölünebilir ve bu da riskli kısımları daha önce geliştirme imkânı sağlar kısaca daha iyi risk yönetimi sağlar. Prototiplerin kapsamlı bir şekilde kullanılmasına olanak sağlar.

**Dezavantajları**

Yönetimi ve süreç daha karmaşıktır. Küçük veya düşük riskli projeler için uygun değildir ve küçük projeler için pahalı olabilir. Çok sayıda ara aşama ve dokümantasyon gerektirir. Projenin sonu başlangıçta bilinmeyebilir.

**Artımsal Geliştirme Modeli**

Bu modelde proje parçalara bölünür ve kullanıcının önceliğine göre bu parçalar sıralanır. Sıralanan bu parçalar bittiğinde sırasıyla birer ara ürün geliştirilmiş olur ve bu ara ürünler de kullanıcı tarafından kullanılır. Ara ürünler her zaman bir öncekinin üstüne bir şeyler katarak ilerler. Yani her bir aşama yeni bir aşama sayılabilir.



Yukarıdaki şekilde şekil sırayla eklenerek devam eder ama bir önceki şeklin tamamlanması beklenir. Bu nedenle şekil bitene kadar eklemeye devam edilir. Şekilde olduğu gibi kullanıcı projeyi düşünür ardından projeyi yapmaya başlar. Proje adım adım tamamlanmaya başlar.

**Artımsal Geliştirme Modelinin Avantajları ve Dezavantajları**

**Avantajları**

Bu model ile sistemin başarısız olma olasılığı azalır, ara ürünler yazılımın geliştirilmesinde önemli bir yere sahip olur. Sonuçlar erken ve sıralı bir şekilde elde edilir. Daha küçük yineleme sırasında sınama ve hata ayıklama kolaydır. Risk yönetimi kolaydır önce riskli parçalar yapılır. Yaşam döngüsü sırasında müşterinin değerlendirmesini ve geri bildirimini kolaylaştıran yazılımlar erken üretilir.

**Dezavantajları**

Her bir parçanın kendi içinde tekrar etmesine izin verilmez bu yüzden de bir ara ürünün bitip diğeri başlayana kadar herhangi bir değişiklik yapılamaz, parçaları oluşturmak için de bu sistemin detaylı bir şekilde tanımlanması lazımdır. Değişim maliyeti daha düşük olmakla birlikte değişen gereksinimler için çok da uygun değildir. Küçük projeler için uygun değildir. Projenin ilerlemesi risk analizi aşamasına oldukça bağlıdır.

**Kodla ve Düzelt Modeli**

Küçük programlar için kullanılır, direkt ürün gerçekleştirilir ve emeklilik safhası vardır. Büyük projeler için kullanılmaz, bakım az vardır ama zordur, ürünü hazırlanıp kullanıma sunulur. Bu model genellikle resmi olmayan bir ürün fikriyle başlar ve program ürün hazır olana kadar ya da gerekli zaman bitene kadar kodlama yapılarak devam eder.

**Kodla ve Düzelt Modelinin Avantajları ve Dezavantajları**

**Avantajları**

Herhangi bir planlamaya ihtiyaç duyulmaz. Proje aşamaları çabuk geçilir. Uzman görüşüne ihtiyaç düşüktür herkes bu modeli kullanabilir. Küçük projelerde kullanılır.

**Dezavantajları**

Kaynak planlaması yoktur bu nedenle hataların bulunması ve doğrulanması zordur. Kodlar kullanıcının ihtiyacını karşılamayabilir. Kodlar sonradan değiştirmek için planlanmadığından esnek değildir, değiştirilmesi zordur​​.

**Evrimsel Geliştirme Modeli**

İlk tam ölçekli modeldir. Geniş alana sahip büyük firmalar için önerilir. Her aşamada üretilen ürün tam işlevselliğe sahiptir. Modelin başarısı ilk evrimin başarısına bağlıdır.

**Çevik (agile) Modeller**

Çevik model, yazılım sistemlerini etkili ve verimli bir şekilde modellemeye verilen addır. Fonksiyonlar arası ekipler, bir ürünün iterasyonları üzerinde belirli bir süre çalışırlar ve bu iş, iş veya müşteri değerine göre önceliklendirilen bir yığılma olarak düzenlenir. Her yinelemenin amacı, çalışan bir ürün üretmektir. Çevik modelde liderlik ekip çalışması, hesap verebilirliği, yüz yüze iletişim ön plandadır. Çevik modelin kullanılmasına en uygun olduğu durumlar şunlardır: projenin yazılım evresinde müşteriden gelebilecek talep değişikliklerinin tahmin edilemez olması, proje parçalarının önce tasarlanıp ardından hemen geliştirmesinin gerekmesi ve önceden ne yapılacağını, detaylı yol haritasını ve tasarımını tahmin etmenin zor olması, analiz, tasarım ve test etme süreçlerinin ne kadar zaman alacağının önceden bilinmemesi, yazılım ekibinin birlikte çalışmak, hiyerarşiye önem vermemek, sağlam iletişim kurmak gibi özelliklere sahip olması.

**Çevik Modelin Avantajları ve Dezavantajları**

**Avantajları**

Değişim ve kolayca uyum sağlama esasına dayandığı için küçük ve orta çaplı projeler için uygundur. Kaynak gereksinimleri çok azdır. Değişen ortamlar ve sabit veya değişen gereksinimler için uygun bir modeldir. Genel olarak planlanmış bir ortamda aynı anda geliştirme ve teslimat yapılmasını sağlar. Yönetmesi kolaydır ve geliştiricilere esneklik kazandırır.

**Dezavantajları**

Karmaşık bağımlılıkların kullanımı için uygun değildir. Sürdürülebilir olması için fazla risk vardır. Müşteri ile etkileşim yoksa proje ekibi yanlış yönde projeye devam edebilir. Dokümantasyon minimum seviyede oluşturulduğu için bireysel bağımlılık çok yüksektir. Dokümantasyon eksikliği yüzünden sonradan gelen yeni ekip üyelerini projeye dahil etmek zor olabilir.

**Extreme Programming (sınırsal programlama)-XP**

XP, 3 kişi tarafından ortaya atılmıştır. Bu kişiler Ward Cunningham, Kent Beck, Ron Jeffries’dir. Günümüzde geniş bir uygulama alanına sahiptir. Sınırsal programlamanın en önemli özelliklerinden bir tanesi, sahip olduğu pratikler hakkında uygulayıcılardan sürekli geri dönüşümlerin olmasıdır. Bu sayede sürekli gelişip iyileşebilir. Sınırsal programlama basitlik, haberleşme, geribildirim, cesaret ve saygı temelleri üzerine kurulmuş olan bir yazılım geliştirme metodudur. Takım üyeleri basit pratikler sayesinde nelerin nasıl yapılması gerektiğini anlarlar. Direkt müşteriden, takım üyelerinden ve testlerden alınan yeterli geribildirim sayesinde takım işin neresinde olduğunu görür. Pratikleri kendisine uygun bir şekilde ayarlamaya çalışır.

**Scrum**

Scrum, 1990’ların başında Jeff Sutherland ve geliştirme ekibi tarafından tasarlanan çevik bir yazılım geliştirme yöntemidir. Kompleks işlerin yönetiminde kullanılmak için oluşturulmuş ve verimliliği değerin üstünde tutan geleneksel metodolojilerden ayrılmış bir yaklaşımdır. Scrum modeli, Scrum Takımlarını ve bu takımların rollerini, etkinliklerini, yarattıklarını ve kuralları kapsar. Scrum modelindeki tüm bileşenlerin özel ve belirli hedefleri vardır ve bu nedenle modeldeki kurallara uyulması başarılı sonuç elde edebilmek için zorunludur. Kısaca Scrum, çalışanların karmaşık sorunları doğru biçimde ele alırken, bir yandan da yüksek değere sahip üretim yapmaları hedefleyen bir yöntemdir. Anlaması ve uygulaması kolay olan bu popüler metodoloji, her ne kadar basit gözükse de ustalaşması zordur.

**Scrum özellikleri:** Bu yöntem yazılım geliştirme aşamasında yöneten ve kontrol eden çevik bir süreçtir. Gereksinimler müşterinin talebiyle hızlı bir şekilde değişirse kademeli olarak sistemi ve yazılımı geliştiren proje ekibi döngüsel bir yaklaşımla değişime ayak uydurur. Proje ekibi arasında iletişimi arttırır ve aralarındaki iş birliğini en üst seviyeye çıkarır.

**KAYNAKÇA**

[**https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742**](https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742)

[**https://webdosya.csb.gov.tr/db/cbs/icerikler/salihsoylu\_tez\_v10-20180925134450.pdf**](https://webdosya.csb.gov.tr/db/cbs/icerikler/salihsoylu_tez_v10-20180925134450.pdf)

[**https://www.codex.com.tr/yazilim-gelistirme-modelleri**](https://www.codex.com.tr/yazilim-gelistirme-modelleri)

[**https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/**](https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/)

[**https://bilginc.com/tr/blog/184/scrum-nedir**](https://bilginc.com/tr/blog/184/scrum-nedir)

**Açılması Gereken Hesaplar**

[**https://medium.com/@sefa4019**](https://medium.com/@sefa4019)

[**https://github.com/sefavnl**](https://github.com/sefavnl)

[**www.linkedin.com/in/sefa-vanlı-769520233**](http://www.linkedin.com/in/sefa-vanlı-769520233)

SEFA VANLI

210601003

BİLGİSAYAR MÜHENDİSLİĞİ