metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**İZMİR BAKIRÇAY ÜNİVERSİTESİ**

**MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**YAZILIM MÜHENDİSLİĞİ TEMELLERİ ÖDEV SUNUMU BAHAR DÖNEMİ**

**HAZIRLAYAN**

**SEBAHATTİN SAFA KARA**

**210603044**

**DR. ÖĞR. ÜYESİ Zekeriya Anıl GÜVEN**

**Yazılım Yaşam Döngü Modelleri**

**Yazılım Nedir?**

Yazılım, elektronik aygıtların belirli bir işi yapmasını sağlayan programların tümüne verilen isimdir. Bir başka deyişle, var olan bir problemi çözmek amacıyla bilgisayar dili kullanılarak oluşturulmuş anlamlı anlatımlar bütünüdür. Yazılım için çeşitli diller mevcuttur. Yazılım günümüzde aklınıza gelebilecek tüm sektörlerde kullanılmaktadır

**Yazılım Yaşam Döngüsü (Software Development Life Cycle, SDLC)**

Yazılım yaşam döngüsü, bir yazılım ürününün oluşturulması ve bakımı da dahil olmak üzere yapılan adımlar topluluğuna denir. Yazılım yaşam döngüsü doğrusal veya tek yönlü değildir bazı temel aşamalardan oluşur. Bu aşamalar; Planlama, Analiz, Tasarım, Gerçekleştirme, Teslim ve Bakım, Test,Gereksinimlerin Belirlenmesi. Kısaca bu aşamalardan bahsedecek olursak

**1)Planlama:** Planlama da yazılım için temel ihtiyaçlar, toplantı tarihleri ve projenin genel anlamda planlanması ile bu basamak tamamlanmış olur. Bu aşama da yazılım ekibi ve müşteri ortak kararlar alarak ileri ki aşamaların daha kolay geçmesini sağlar.

**2)Gereksinimlerin Belirlenmesi:** Müşterinin istekleri ve temel ihtiyaçları belirlenmeye çalışılır. Sonrasında müşteriye gereksinim önerileri verilir.

**3)Analiz:** Planlama aşamasında belirlenen ihtiyaçlar ve gereksinimler netleştirilir, ardından belgeye dökülür. Bu netleştirme işlemi için yazılım mühendisiyle müşterinin iletişimi güçlü ve sağlıklı olmalıdır

**4)Tasarım:** Bu aşamada gereksinim ve analiz aşamasında toplanan veriler üzerinden girdi sağlanarak sistemi oluşturacak olan yazılım mimarisinin ortaya çıkarıldığı, türetildiği aşamadır.  **5)Gerçekleştirime:** Bu aşamada önceki aşamalarda belirlenenlere göre müşteriye sunulacak olan ürünün kodlaması yapılır. Kod “Clean Code” felsefesine uygun olacak biçimde kısa, net ve anlaşılır biçimde yazılır. Temiz kod yazmamız hem yazılımı hazırlayan bize hem de sonrasında yazılımı inceleyecek olan yazılımcılara kolaylık sağlayacaktır.

**6)Test:** Sisteme ait her modül için kodlar yazılır. Daha sonrasında her modül birbirinden bağımsız olacak şekilde testler yapılır. Oluşturulan yazılımın çalışabilmesi için tüm modüller birleştirilir. Modüllerin birleştirilmesi sırasında ortaya bazı hatalar çıkabilir. Bu gibi durumlar test edilir. Hem kodlama süresince hem de kodlama sonrasında yapılması gereken en önemli şeylerden biri testtir. Analiz aşamasından itibaren her aşamada testler yapılmalıdır. Hataların ortaya çıktığı aşama, teslim aşamasına ne kadar yakın olursa hataların düzeltilme maliyeti de düzeltilme süresi de o kadar artar. Bu yüzden erken test çok önemlidir. Yazılımcıya zaman, para ve prestij kazandırır. Gerçekleştirme aşamasında sistem artık hayata geçer. Bu aşamada artık gerçek veri kullanılarak sistem test edilir. Bu test kabul testidir ve müşteri tarafından yapılır.

**7)Teslim ve Bakım:** Yazılım oluşturulup müşterinin hizmetine sunulduğu aşamadır. Teslim ürünü olarak yalnızca yazılım ürünü yeterli değildir. Bunun yanında ayrıca kullanıcılar için kullanım kılavuzu da olmalıdır. Ürün teslim edildikten sonra zaman içinde hatalarının giderilmesi veya güncellenmesi gerekebilir. Bu sürekli devam edecek olan bir aşamadır. Bu noktada müşteri geri dönüşleri büyük önem taşımaktadır. Bu süreçte iki tarafın uzlaşması ve onayıyla yazılıma yeni özellikler eklenip çıkarılması da yapılabilir.

Yazılım yaşam döngüsünde birden fazla model vardır. Birden fazla olmasının sebebi ise projelerin büyüklük farklılıkları ve hazırlandığı kişi ve kitlelerin farklılığı gibi sebeplerdir. Yazılım sektöründe en çok kullanılan modeller;

**1- Gelişigüzel Model**

1960’lı yıllarda kullanılan bu yöntem genellikle tek bir kişinin üretim yaptığı küçük ve basit projelerde kullanılır. Bu modelde herhangi bir sistem veya yöntem yoktur. Bu yüzden tam bir model olarak ele alınmaz. Kodlama tarzı ve süreç kişiden kişiye değişkenlik gösterir. Aynı zamanda dokümantasyonda yapılmaz. Bu sebeplerden dolayı yazılımın güncellenmesi ve bakımı çok zor hale gelir hatta bazen güncelleme ve bakım yapılamayabilir.

**2- Barok Modeli**

Bu modelde, dokümantasyon ayrı bir süreç olarak ele alınır. Bu günümüzde kullandıklarımızdan farklıdır. Bu yüzden günümüzde uygulanan bir model değildir. 1970’li yıllarda ortaya çıkmıştır.

**3- Şelale (Waterfall) Modeli**

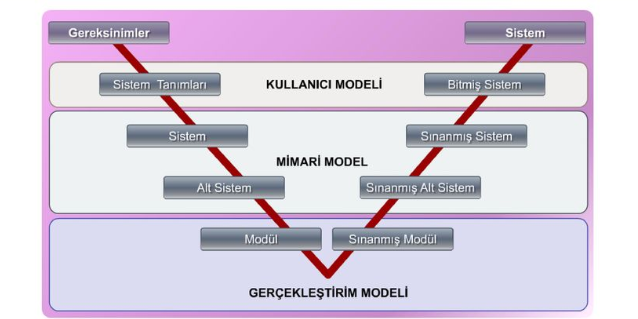
diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Çağlayan model yaklaşımında öne sürdüğümüz temel adımlar sırasıyla gerçekleştirilir; herhangi bir adımda sorun çıkması halinde o adıma geri dönülerek düzeltilir. Genel kapsamıyla yazılım projelerinin ilerleyişi için kullanılması temel alınmış bir modeldir.Gereksinim tespitini tasarım sürecinin takip etmesi yönünden; “Yap düzelt modele” göre daha komplekstir. Çağlayan modelin eksik yönlerini tespit etmek istersek; yazılım sürecinde iken müşteri ile olan iletişim azlığını öne sürebiliriz. Uygulama esnasında müşteri ile etkileşimde kalınmadığı için somut müşteri geri dönütleri ancak proje tamamlandığında alınabilmektedir.

**4- V Süreç Modeli**

Bu modele şelale modelinin gelişmiş hali denilebilir. V süreç modelinin sol tarafı üretim, sağ tarafı test işlemlerini gösterir. Bu modelin aslında 3 temel çıktısı vardır. Bunlar; Kullanıcı Modeli, Mimari Model, Gerçekleştirim Modelidir. Kullanıcı Modelin de kullanıcının istekleri, proje için tanımları kullanılır ve planlar yapılır. Mimari Model de projenin tasarımı ve tüm sistemin test işlemleri yapılır. Gerçekleştirim Modeli ise yazılımın kodlarının yazılması ve test işlemleri yapılır. Bu modelin en büyük avantajı takibinin ve kullanımının kolay olmasıdır. Bu model belirsizliklerin az olduğu aynı zamanda iş tanımlarının net bir şekilde belirlendiği bilgi teknolojileri projeler için uygundur. Bu modelin en büyük sorunları aşamalarda tekrarların olmaması ve risk çözümleme olmamasıdır.



**5- Helezonik (Spiral) Model**

çizelge, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Helezonik modeli risk analizinin ön planda olması sayesinde hataları erken çözme olanağı olur. Prototip oluşturma her aşama da gerçekleştiği için kullanıcı da her aşamada yazılım projesinin bir parçasını görür ve bu da sorunların azalmasına yardımcı olur. Planlama aşamasında her aşamada olan ara ürün için planlama ve amaç belirleme yapılır. Risk Analizinde ise riskler araştırılır, belirlenir ve çözülür. Üretim aşamasında ara ürün üretilir. Kullanıcı Değerlendirme aşamasında ise kullanıcıya sunulan ara ürünün kullanıcı tarafından sınandığı ve değerlendirildiği aşamadır. Bu modelin en büyük sorunları küçük ve düşük ölçekli projeler için maliyetli olmasıdır. Daha çok büyük ölçekteki projeler için kullanılmalıdır. Helezonik model karmaşıktır. Ara adımların çok fazla olması sebebiyle oldukça fazla dokümantasyon gerektirir.

**6- Artımsal Geliştirme Süreç Modeli**

Artışlı model genel kapsamıyla çağlayan modele benzerlik gösterir; bu modeli çağlayan modelden ayıran unsur yazılım geliştiricilerinin öncelikle müşterinin öncelikli gereksinimlerini baz alıp uygulamaya koymasıdır. Her bir yapı bitiminde yeni yapıya geçildiği gibi önceki yapının da bakımı yapılır. Bu durum müşterilerin de istediği bir yaklaşımdır. Müşteri bu sayede daha az önemli olan fonksiyonların uygulanması için beklemek zorunda kalmaz. Bu modelin olumsuz kısımlarını değerlendirmek istersek; modelin her proje için uygun olmadığını görürüz. Artışlı model yaklaşımı istenilen fonksiyonların modüler olarak ayrılabilen tipte olmasını gerektirir. Ne yazık ki farklı bir fonksiyona bağlı olmadan görevini yerine getirebilen fonksiyonların sayısı oldukça azdır. Bu durumda da modülerlik oluşması beklenemez

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**7- Kodla ve Düzelt Yaşam Döngü Modeli**

Herhangi bir planlama yapılmaz. Çok küçük projelerde kullanılır. Program aşamaları hızlı geçilir. Diğer modellerden farklı olarak emeklilik safhası içerir. Bu modelin sorunlarına gelirsek; planlama yoktur, bitiş süresi net değildir, hataların bulunması zordur, kodları değiştirmek zor ve maliyetlidir

diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

**8-Evrimsel Geliştirme Modeli**

Coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir.

Bu model ilk tam ölçekli modeldir. Büyük alanlara yayılmış bir modeldir. Büyük firmalar için önerilir. Modelin başarısı ilk evrimin başarısına bağlıdır.

**Çevik Modeller**

Bu modeller başarı oranını ve ekip içi iletişimi artırmak için ortaya çıkmıştır. Bazı çevik modeller şunlardır;

**1-Extreme Programming (XP)**

Kent Beck ve arkadaşları tarafından 1996 yılında kurulmuştur.4 temelden oluşur: Basitlik, Cesaret, Geri Dönüş, İletişim.

**Basitlik:** Yapılan işin sade ve anlaşılır yapılmasını gerektirir.

**Cesaret:** Projelerin üstüne korkmadan ilerlenmelidir. Gerekirse bütün kodun silinip yeniden yazılması sağlanır.

**Geri Dönüş:** Geri dönüşlerle yapılabilecek hatalar hatalar azaltılır. Müşteri ile yazılım ekibi birbiriyle iletişim halindedir.

**İletişim:** Projelerdeki önemli sorunlardan biridir. XP bu sorunu çözmek için çalışır ve ekip içi iletişime önem verir artırılması için çalışır.

**2-SCRUM**

Scrum programlama yaklaşımı kompleks ve ilerisi tam anlamıyla görülemeyen projelerde uygulanır. Gerçekleştirilmesi istenen bütünün gereksinim önceliklerine göre küçük parçalara (sprint) ayırılmasını benimser (product backlog). Bu küçük parçalar 30 gün içerisinde halledilebilir parçalar olmalıdırlar (sprint backlog). Parçalar gerçekleştirilirken yazılım ekibi günde 15 dakikalık toplantılar yaparak yazılımın genel kapsamıyla ilgili birbirleriyle iletişim halinde olurlar. Bu 15 dakikalık toplantılar projenin sağlıklı ilerletilebilmesi için oldukça önemlidir (sprint backlog). Scrum programlama yaklaşımının günümüzde bu kadar popüler olmasının nedeni yapılması istenen işin kısa süreler içerisinde halledilebilir olmasını vaat ediyor oluşudur. Scrum programlama yaklaşımının temel kavramları; roller, toplantılar ve bileşenler/araçlardan oluşur.

**1-Roller**

Müşteri projenin ilerleyişinde yazılım ekibine geri dönütler vererek istenilen şeklini almasından sorumludur. Scrum yöneticisi ekibin yazılım ortamını belirler. Yazılımın ileri safhalarında ekibin nerelerde zorlandığını gözlemleyerek ekibe destek olur. Scrum takımı yazılımı gerçekleştirecek bireyler topluluğudur.

**2-Toplantılar**

Sprint planlama toplantıları genel olarak bir yazılıma başlanmadan önce yapılan gereksinimlerin belirlenmesi, yapılacakların küçük kısımlara ayrılması, küçük kısımların ekip üyelerine dağıtılması, olası ilerde karşılaşılacak hataların tespiti, yazılımın hangi sistem ve araçlar kullanılarak yapılacağı gibi temel problemlerin karşılaştırıldığı toplantılardır. Sprint gözden geçirme toplantıları planlama toplantılarında elde edilen verinin planlanıp uygulamaya geçilmesi süreçlerinin konuşulduğu toplantılardır. Günlük scrum toplantıları ekip içerisinde geri dönüşlerin alınmasını sağlar. Günde 15 dakika boyunca, o gün içinde ne kadar ilerleme kaydedilebileceği, oluşan sorunlara yönelik neler yapılabilineceği gibi yazılım sürecinin doğal olarak getirdiği durumlar tartışılır.

**3-Bileşenler/Araçlar**

Ürün gereksinim dokümanı müşterinin yazılımdan beklediği kısımların belgelenmiş halidir. Bu belgeye ihtiyaca göre yeni gereksinimler eklenebilir. Aynı zaman gereksiz olduğu anlaşılan (müşterinin geri dönüşüyle) kısımlar kaldırılarak sadeleştirilebilir. Sprint dokümanı yapılması kararlaştırılan işi ve ekip içi görev dağılımını belgeler. Sprint kalan zaman grafiği ekip üyelerinin bir iş (sprint) için tanımlanan süre ile gerçekleştirdiği süreyi girdiği belgelerdir. Ekip üyeleri bu belgeye bakarak kalan işi ve zamanı daha iyi gözlemleyip bu gözleme göre kalan zamanı planlamadırlar.

**Yazılım Geliştirme Modellerinin Karşılaştırılması**

Şelale modelinin kullanılması ve uygulanması basit, iş bölümü ve iş planlaması detaylıdır. Küçük ve özellikleri belli projelerde kullanılır.eskiden yaygın olarak kullanılıyordu ancak iletişim yetersizliğinden dolayı, büyük projelerde yetersiz olması ve değişimlere kapalı olması nedeniyle günümüzde çok kullanılmıyor. V modeli şelale modelinin gelişmiş versiyonu diyebiliriz. Kullanımı ve takibi kolay olması, onaylama ve doğrulama işlemlerinin planın erken aşamasında uygulanabilir olması avantajlıdır ancak fazlar arası tekrarlamalara imkân tanımaması, risk çözümleme basamaklarının bulunmaması gibi sorunlar bulundurur.Spiral modelde kullanıcılar sistemin içerisindeler. Risk analizi iyidir. Büyük projeler için idealdir. Hatalar erken giderilir. Ancak karışıktır ve çok üzün sürer.Artımsal Geliştirme modelinde müşteri her seferinde ürünü gördüğü için işlevseldir proje daha güvenlidir. Ancak her artımı tamamlamak için sistemin detaylı olarak tamamlanması lazımdır ve kendi içlerinde tekrarlanmazlarGelişigüzel ve barok modelleri yinelemeli olmadıkları ve dökümantasyonu işiniçinde tutmadıkları için günümüzde kullanılmıyan modellerdir.

**Hangi Projede Hangi Modeli Kullanmalıyız?**

Küçük ve özellikleri iyi tanımlanmış projelerde şelale modelini kullanmak uygundur.Büyük maliyetli ve uzun süren projelerde spiral ve artımsal geliştirme modelini kullanmak uygundur.Orta ve küçük, uzun sürmeyen projelerde çevik modelleri kullanmak uygundur.

**Scrum Günümüzde Neden Popüler?**

Kararları vermekten sorumlu olan ekip, onları her zaman tetikte güncellemeler ile birlikte ilerler. İş gereksinimi belgesi önemsiz kabul edilir. Zaman ve paradan büyük miktarda tasarruf edilir. İletişim ve takım çalışması önemli olduğundan hatalar erkenden fark edilip düzeltilir. Karmaşık projelerde etkilidir. Günlük toplantılar geliştiricinin bireysel verimliliğini ölçmesine yardımcı olur. Bu durum, ekip üyelerinin verimliliklerinin artmasına yardımcı olur. Çok esnek olduğundan hemen her değişime uyum sağlayabilir.

**KAYNAKÇA**

* DR. ÖĞR. ÜYESİ Zekeriya Anıl GÜVEN, Bakırçay Üniversitesi Yazılım Mühendisliği Temelleri Dersi 2. Ve 3. Hafta Sunumları
* <https://phoenixnap.com/blog/software-development-life-cycle>
* <https://www.professionaldevelopment.ie/who-uses-scrum>
* <https://clarusway.com/what-is-software-development-life-cycle/>
* <https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>
* <https://www.hosting.com.tr/blog/github-nedir/>
* <https://www.brandingturkiye.com/medium-nedir-nasil-kullanilir-niye-onemlidir/>