**YAZILIM YAŞAM DÖNGÜSÜ VE MODELLERİ**

Elif Rümeysa TURAN

İzmir Bakırçay Üniversitesi

Bilgisayar Mühendisliği Bölümü

210601064

Özet

Yazılım, günümüzde dünyadaki her kesim insanın ihtiyacını duyduğu, olmazsa olmazları haline gelmiş bir programdır. İnsanların her türlü istek ve ihtiyaçlarını karşılamada her geçen gün bir adım daha ileriye gitmiştir. İhtiyaçları karşılamada daha kaliteli ürünler elde edebilmek için birçok yola başvurmuşlardır. Bu makalede kaliteli ürün elde etmek için kullanılan Yazılım Yaşam Döngüsü ve Modellerinden bahsedilmiştir.

1. GİRİŞ

Yazılım, günümüzde dünyadaki her kesimden insanın artık ihtiyacını duyduğu ve olmazsa olmazları haline gelmiş bir çeşit programdır. İnsanlığın ihtiyaçlarını karşılamada en büyük etkenlerden biri olan yazılım, oldukça titiz bir çalışma ister. Sürekli yenilenebilir ve gereksinimleri değişebilir olan yazılımı en doğru ve güvenilir bir şekilde yerine getirmek için yazılım proje yönetiminin oldukça iyi olması gerekir. Proje yönetimi ne kadar iyiyse planlamalar kolaylıkla yerine getirilir, yazılım beklenilenden daha kaliteli bir hale getirilebilir.

Proje yönetiminin başlıca görevlerinden biri müşterinin gereksinimini iyi anlamaktır. Müşteri gereksinimi ne kadar iyi anlaşılırsa hata yapma oranı da bir o kadar düşer. Yazılım, müşterinin gereksinimlerine göre şekilleneceği için en ufak detay dahi yazılımın kalitesini değiştirebilecek güce sahiptir. Yazılım sürekli değişen, gelişen ve esnek bir yapıya sahiptir. Üretilen, üretim aşamasında olan yazılımın aynı kalması beklenilmez. Günler geçtikçe teknolojinin getirdiği ihtiyaçların artmasıyla müşterilerin de ihtiyaçları arttığı için üretilen veya üretim aşamasında olan yazılımın da değiştirilmesi gerekir. Bundan dolayı değişken ve esnek olan bir plana ihtiyaç duyulur. Proje yönetiminin bu değişkenliklere, yeni gereksinimlere açık olması ve ona göre plan kurması beklenir. Plan kurulurken belirlenen zamanı aşmak veya belirlenmiş olan bütçeyi aşmak müşteriyi hoşnut etmez. Bundan dolayı proje yönetimi plan aşamasında oldukça titiz davranmalıdır.

1. YAZILIM YAŞAM DÖNGÜLERİ

Yazılım üretim aşamasında meydana gelebilecek her türlü değişkenliklere ve yeni gereksinimlere karşı esnektir. Proje yönetiminin işini kolaylaştıracak olan yazılım yaşam döngüleri birçok yazılım üretilirken kullanılır. Software Development Life Cycle (SDLC) olarak bilinir. Yazılım yaşam döngüsü, üretilecek yazılımın neredeyse her aşamasında yer alır. Bu döngü tek bir yöne ilerleyen doğrusal bir döngü değildir, aksine kendisini her yöne yönlendirebilen, her türlü değişime açık olan ve istenildiğinde önceki aşamaya rahatlıkla geçmesini sağlayan bir döngüdür. Bundandır ki yazılım üretilirken kendini sürekli yineleyebilir, tekrar tekrar kontrol ederek değişiklikler yapabilir. Yazılım yaşam döngüsü birçok aşamadan meydana gelir.

2.1 YAZILIM YAŞAM DÖNGÜSÜ AŞAMALARI

Yazılım yaşam döngüsü genel olarak 6 aşamadan oluşmaktadır. Bunlar; planlama, analiz, tasarım, gerçekleştirme, teslim ve bakım aşamalarıdır.

* + 1. Planlama:

Bu aşama yazılım yaşam döngüsünün ilk aşamasıdır. Proje için ‘ne istiyoruz?’ sorusu üzerine odaklıdır. İstenilen hedefe ulaşmak için bu adımda müşterinin gereksinimlerini oldukça iyi dinlemek, bunları tam ve doğru şekilde anlamak gerekir. Gereksinimler anlaşıldıktan sonra bu gereksinimlerin nasıl yerine getirileceği, hangi donanım gereçlerine ihtiyaç duyulacağı, hangi personellerden yardım alınacağı, hangi yolların kullanılacağı gibi ihtiyaçların listesi çıkartılır. Daha sonra projenin fizibilitesi belirlenerek projenin başarılı olup olmayacağı, yapılıp yapılamayacağı sorgulanır. Bir projenin en büyük amaçlarından biri üst yönetimin belirlediği sürede ve kalitede bir yazılım üretmektir. İhtiyaçlar belirlendikten sonra belirlenmesi gereken en büyük faktörlerden biri hedef kitledir. Üretilecek yazılımın hangi hedef kitleye hitap edeceği çok önemlidir. Üretilen yazılımın en büyük amaçlarından bir diğeri ise müşteri memnuniyetidir. Müşteri ne kadar memnun olursa oluşturulan yazılım bir o kadar kaliteli demektir.

* + 1. Analiz:

Bu aşamada planlama aşamasında belirlenen ihtiyaçlar daha ayrıntılı bir şekilde analiz edilir ve herhangi bir belirsizlik olup olmadığı kontrol edilir. Herhangi bir belirsizlik olduğu fark edilirse bir önceki aşamaya geri dönülerek müşteri ile tekrar irtibata geçilip, gereksinimler üzerinden tekrar geçirilerek yeniden ihtiyaç listesi oluşturulur. Yeni liste tekrar analiz edildikten sonra herhangi bir belirsizlik yok ise bu ihtiyaçlar ve analizler belirli bir formatta belge haline getirilir. Bu belgenin proje geliştirme sürecini olumlu etkilemesi için üretim aşamasında yer alan geliştiriciler tarafından iyi anlaşılması gerekir. Hazırlanan bu belgenin müşteri ile sürekli etkileşim halinde olması ve kendini yineleyebilmesi için müşterinin de anlayacağı formatta olması gerekir. Bu aşamanın bir diğer özelliği ise temel UML diyagramlarının çizilmeye başlandığı ilk aşama olmasıdır.

* + 1. Tasarım:

Birinci ve ikinci aşamalar olan planlama ve analizde belirlenen gereksinimler yerine getirilerek gerçekleştirme aşamasından önce bir model tasarlanır. Bu model tasarlanırken UML, nesne modeli veya iş şeması gibi daha birçok yöntemlerden herhangi biri kullanılabilmektedir. Bu aşamada kod yazılmaz. Tasarım aşamasının en önemli tekniklerinden biri Soyutlama (Abstraction) tekniğidir. Bu teknik planlama aşamasından önce belirlenen problemi daha basit hale indirgeyerek problemdeki daha önemli olan kısımların dikkate alınmasına kolaylık sağlar. Problem içerisinde yer alan olayların ve durumların bazı özelliklerinin görmezden gelinmesini sağlar.

Tasarım aşaması iki önemli aşamadan meydana gelir. Bu aşamalar mimari (sistem) tasarımı ve ayrıntılı tasarımdır. Mimari tasarım öncelikle genel sistem mimarisini kurar. Mimari sistem, genel sistemin soyut bir ifadesini temsil eder. Mimari tasarım, oluşturulmuş olan genel sistemin oluşturulacak ürünün gereksinimini karşılayıp karşılayamadığını kontrol ederek, gelecekte meydana gelebilecek gereksinimlerin karşılanabilmesi ile ilgilenir. Tüm bu işlemlerin sonunda elde edilen veriler doküman haline getirilir. Ayrıntılı tasarım ise mimari tasarımdaki dokümanların tekrar ele alınması ve incelenmesinden meydana gelir. Yazılacak yazılımı titiz ve tam bir şekilde ifade etmeye çalışır. Yazılımı içeren bileşenler ve bunların ayrıntısını içerir.

* + 1. Gerçekleştirme:

Bu aşama kodlama ve test etme aşaması olarak ikiye ayrılır. Kodlama aşaması, planlama, analiz ve tasarım aşaması sonucunda elde ettiğimiz verilere göre ürünün programlanması aşamasıdır. Bu aşamada analiz aşamasında belirlenen teknolojilerden yararlanılır. Kodlama aşamasında projeye ait ilk çıktılar meydana gelir.

Test etme aşaması kodlama aşaması sonucu meydana getirilen ürünün kontrol edilmesi aşamasıdır. Yazılımcılar kendilerine ‘planladığımızı elde edebildik mi?’ sorusunu sorarak ürünün istenileni karşılayıp karşılayamadığı kontrol edilir. Test aşamasının en önemli noktalarından biri ürünün kodlama aşamasındayken sık sık kontrol edilmesidir. Bu sayede hata yapma oranı en aza indirilerek maliyet bakımından yarar sağlanmaya çalışılır. Hata oranı azaldıkça maliyet de düşecektir. Ürünün test aşamasında kontrolü sırasında fark edilen gereksinim eksikleri veya hatalar düzeltilmek için eski aşamalara geri dönülmesi gerekir. Bir sorun yoksa diğer aşamaya geçilebilir.

* + 1. Teslim:

Test aşamaları tamamlandıktan sonra hazır hale gelmiş yazılımda herhangi bir sorun yoksa son dokunuşlar yapılarak teslim edilebilir hale getirilir ve hedef ortamına kurulur. Bu işlemler yapılırken kullanıcıya yalnızca yazılımı teslim etmek yeterli gelmez, yanında yazılımın kullanım kılavuzu ve dokümanları da bulunmalıdır. Yazılımın nasıl kullanılacağına dair eğitimler vermek ürünün kullanılabilirliğini artırmak ve ömrünü uzatmak açısından önemlidir.

* + 1. Bakım:

Kullanıcılar ürünü kullandıkça ve kullanmaya devam ettikçe gelişen teknoloji ve gerek duyulan ihtiyaç artımı sonucunda birçok soru ve sorunlarla karşı karşıya kalacaktır. Bu yüzden hedef ortama kurulan yazılımla sürekli iletişim halinde olmak ve yazılıma destek sağlamaya devam etmek çok önemlidir. İletişim halinde oldukça ürünün sorunları hemen fark edilir ve düzeltmek için hem zamandan hem de bütçeden yarar sağlanır. Gereksinimler arttıkça oluşacak sorunları düzeltmek, yazılımın kurulumun aşamalarında harcanan zamandan çok daha fazla zaman alabilir.

Yazılım yaşam döngüsü model çeşidi sayısı çok fazladır. İhtiyaç duyulan yazılıma göre model seçilebilir, proje yönetiminin işi daha da kolay hale getirilebilir.

1. YAZILIM YAŞAM DÖNGÜSÜ MODELLERİ
   1. Gelişigüzel Model:

Bir model olarak adlandırmak doğru değildir. Herhangi bir yöntem izlenmeden yapılır. Üretim şekli geliştiriciye göre değişir. Bu yüzden takip edilmesi ve bakımının yapılması zordur. 60’lı yıllarda uygulanmıştır ve basit programlama içeren ürünlerde kullanılır.

* 1. Şelale Modeli (Waterfall Model):

1970 yılında Royce tarafından üretilmiş olup, yazılım geliştirme sürecinin ilk modeli olarak varsayılır. Şelale modeli geleneksel bir modeldir, klasikleşmiş adımlardan ve aşamalardan geçer. Bu modelin adımları ardışıktır yani bir sonraki aşamaya geçmeden önce önceki aşamadaki gereksinimlerin tamamının tamamlanmış olması gerekir. Model dinamik değildir, statiktir. Modelde belirsiz gerekliliklere yer verilmez. Modelin her aşaması sonunda bir doküman oluşturulur yani model doküman güdümlü olarak bilinir. Kullanıcı katılımları yalnızca başlangıç aşamasında yer alır, daha sonraki herhangi bir adımda kullanıcı veya müşteriyle iletişime geçilmez.

Şelale modelinin avantajları: Model klasikleşmiş adımlardan oluştuğu için basit ve anlaşılabilirdir. Dezavantajları: Kullanıcı katılımının yalnızca ilk aşamadadır. Bir yazılım ürününe kullanıcı ne kadar katkı sağlarsa o ürün daha kaliteli olur ve hata payı en aza indirilir. Kullanıcı proje içerisinde ne kadar bulunmazsa ürünün kalitesi ve güvenilirliği azalıp, hataya daha elverişli hale gelir. Modelin ardışık aşamalardan oluşur. Aşamalar tamamlanıp diğerine geçildiği için bir hata veya eksik fark edildiğinde veya bir şey eklenmek istenildiğinde geri dönüş sağlanamaz. Başlangıçta yapılan hataların tespitinin uzun zaman alma ihtimali vardır. Hata, yazılımcılar tarafından ne kadar geç fark edilirse ürün için belirlenen bütçe o kadar aşılacaktır, bu da üst yönetim tarafından hoş karşılanmayacaktır.

* 1. V Modeli (V-Şekilli Model):

V modeli Şelale modelinin gelişmiş hali, bir uzantısıdır. Disiplinli bir modeldir ve sonraki aşamasına geçebilmek için bir önceki aşamasını tam anlamıyla tamamlamak gerekir. Her aşamadan sonra doğrulama yapılır. Her yazılımı geliştirme aşaması için bir test aşaması mevcuttur. Her aşamasında test olması hata yapma payını en aza indirir ve herhangi bir gereksinim eksiği veya hata ile karşılaşıldığında kullanıcı ile sürekli iletişim halinde kalınır bu da kullanıcının yazılım üzerindeki rolünü artırarak daha kaliteli ürünlerin üretilmesine yol açar. Modelde V şeklinde bir yol izlenir. Yolun sol kısmı üretim, yolun sağ kısmı ise test aşaması için kullanılır. V modeli belirsizliklerin az, iş tanımlarının yeterince açıklanmış ve belirgin olduğu bilişim alanındaki projeler için uygun bir modeldir.

V modeli iki evreden meydana gelir. Bunlardan birincisi Doğrulama Evresi’dir. Doğrulama evresinin alt dallarından ilki Gereksinim Analizi aşamasıdır. Bu aşamada üretilecek yazılım için gereksinimler titiz bir şekilde toplanır. Toplanan gereksinimlerin sonucunda bir kullanıcı gereksinim dokümanı oluşturulur. Bu doküman kullanıcıya ürünün sistemini anlatmak için iş analistleri tarafından kullanılır. Kullanıcı kabul testi bu aşamada tasarlanmaya başladığı için kullanıcılar bu dokümanı dikkatli bir şekilde incelerler. İkinci alt dal Sistem Tasarımı aşamasıdır. Bu aşamada sistem mühendisleri dokümanları inceleyerek, bir önceki evrede kullanıcılara anlattıkları sistemin nasıl işlediğini anlamaya, yorumlamaya çalışırlar. Müşteri gereksinimlerinden herhangi birinde uyuşmazlık varsa müşteri bilgilendirilir ve hemen çözüm bulunur. Bulunan çözüm sonucunda bir önceki evrede hazırlanan kullanıcı gereksinim dokümanı düzenlenir. Bu aşamada sistem testi için doküman hazırlanmaya başlanır. Üçüncü alt dal Mimari Tasarım aşamasıdır. Bu aşamada hazırlanacak yazılımın mimarisi için seçilen mimaride temel tipik olarak hangi modüllerin içerileceği listesini ve diğer hangi teknoloji detaylı özelliklerden yararlanılacağını dair özellikler sunulur. Bu aşamada entegrasyon test etme tasarımı özel bir evre içerisinde meydana getirilir. Dördüncü ve son alt dal Modül Tasarım aşamasıdır. Modül tasarım düşük seviyeli bir tasarım aşamasıdır. Yazılımın tasarım sistemi çok küçük parçalara ayrılarak her parça için programcıya ayrı ayrı en detaylı haliyle kodlanacak şekilde açıklanır. Hata mesaj listesi, eksiksiz girdi ve çıktılar içerir. Birim test tasarımı bu aşamada gelişmeye başlar.

V modelinin bir diğer evresi Geçerli Kılma Evreleridir. Bu evrelerin ilki Birim Test Etme evresidir. Bu evre kod seviyesindeki hataları kaldırmak için çalıştırılır. İkinci evre Entegrasyon Test Etme evresidir. Oluşturulacak yazılımın sisteminin her bir farklı parçasının birlikte çalışıp çalışmayacağı kontrol edilir. Her testin sonucu da müşteri ile paylaşılır. Üçüncü evre Sistem Test Etme evresidir. Bu evrede oluşturulan yazılımın üst yönetimin, müşterinin ve hedef kullanıcıların beklentilerini karşılayıp karşılayamayacağından emin olunur. Bu yüzden tüm uygulamanın işlevselliği test edilir. Yazılım için belirlenen işlevsel olan ve işlevsel olmayan gereksinimlerin karşılanıp karşılanmadığı sorgulanır, doğrulanır. Dördüncü evre Kullanıcı Kabul Etme evresidir. Üretilen yazılımın gerçek zamanlı, güvenilir ve tam bir şekilde kullanımı için yazılım sisteminin uygulanmasına ve dağıtılmasına hazır olup olmadığına karar verir.

V modelinin avantajları olduğu kadar dezavantajları da vardır. V modelinin avantajlarına disiplinli olması, her aşamanın tamamlanmış olması, basit ve kolay anlaşılır bir model olması, gerekliliklerinin iyi anlaşılan küçük projeler için kullanıma uygun olması, belirli aşama ve süreçlerden geçmesi verilir. Dezavantajlarına ise uzun zamanlı projeler için zayıf bir model olması ve karmaşık yapılı projeler için iyi bir model olmaması örnek verilir.

* 1. Spiral Model:

Bu model ilk olarak Barry Boehm tarafından 1986 tarihli “A Spiral Model of Software Development and Enhancement” makalesinde tanımlanmıştır. Bu makale geniş kitlelere yayılmıştır. Artırımlı, Şelale veya Evrimsel Prototip Oluşturma gibi bir veya daha fazla süreç modelinin ögelerini bünyesinde barındırır. Bu model büyük kapsamlı projeler için uygundur. Spiral modelinin en önemli özelliği risk yönetimi yapabilmesidir. Riskleri projeye başlamadan önce değil sonrasında ele alır. Bu risklerin çözümlerini prototip oluşturarak yapar. Prototipler ile risklerle başa çıkmaya çalışır. Spiralin uzunluğu tam olarak bilinmemektedir ve bu uzunluk projeden projeye değişiklik gösterir. Projenin her bir döngüsüne yazılım geliştirme sürecinin bir aşaması denir ve projeye bağlı olarak bu aşamaların sayısı proje yönetimi tarafından değiştirilir. Bundan dolayı proje yöneticisinin rolü spiral modelde oldukça önemlidir. Spiralin yarıçapı yazılımın maliyetini, açısal boyutu ise bulunduğu aşamada kaydettiği ilerlemeyi temsil eder. Spiral modelin her aşaması dört parçaya bölünmüştür:

* + 1. Hedef Belirlenmesi:

Bu aşamada gereksinimler toplanır, projenin hedefi belirlenir. Belirlenen hedef detaylandırılıp analiz edilir.

* + 1. Risklerin Tanımlanması ve Çözümü:

Belirlenen hedefin en iyi çözümü için çözüm sayılabilecek her ihtimal değerlendirilir. Belirlenen çözümün riskleri belirlenip tanımlandıktan sonra bunu çözebilecek en iyi çözüm yolu bulunarak risk ortadan kaldırılır. Bu aşamadan itibaren prototip oluşturulmaya başlanır.

* + 1. Ürünün Bir Sonraki Sürümünü Geliştirin:

Belirlenen ve tanımlanan özellikler test edilerek geliştirilir, güvenilirliği doğrulanır.

* + 1. Bir Sonraki Aşamayı Gözden Geçirin ve Planlayın:

Müşteri yazılımın geliştirilmiş olan kısmına kadar projeyi değerlendirir. Bir sonraki aşama için planlama başlatır.

Müşteri gelinen kısımdan memnun kalana kadar bir sonraki aşama planlanmaya devam edilir. Müşteri memnuniyeti sonunda planlama ve prototipleme aşamaları durdurulur. Spiral modelin avantajları; gereksinimlerde esneklik sağlanarak değiştirme isteği karşılanır, prototip kullanımına izin verir, risk yönetimi yapılır, büyük projeler için uygundur ve kullanıcılar veya müşteriler sistemi başlangıcında görebilir. Dezavantajları; model karmaşıktır, projenin başlangıcı bilinse de sonu belirsiz hale gelebilir, küçük projeler için uygun değildir.

* 1. Artırımlı Model (Incremental Model):

Artırımlı model ile üretilen her bir yazılım sürümü birbirini kapsayacak ve gittikçe artacak biçimde işlev içerecek şekilde geliştirilir. Bu modelde bir taraftan kullanım, diğer taraftan ise üretim yapılır. Yinelemeli bir modeldir, döngüler kolayca yönetilebilecek küçük parçalara ayrılır. Model bir sürece, takvime bağlı olarak yazılımı kısım kısım geliştirerek teslim etmeye dayanır. İlk başta yazılımın en basit ihtiyaçları karşılanır. Yazılım geliştirme süreci ilerledikçe daha fazla ihtiyaç karşılanır. Süreç ilerledikçe sonunda müşterinin gereksinimleri karşılanır hale gelir. Bu süreçte birden fazla döngü meydana gelebilir ve bu döngüler aynı anda gerçekleşebilme özelliğine sahiptir. Gerçekleşen her bir yeni döngü, yazılım için yeni bir sürüm anlamına gelir ve ürüne yeni özellikler kazandırır. Tekrar kullanılabilecek bir ürün, fonksiyonellik sağlamış bir şekilde tüm döngülerin sonunda ortaya çıkar.

Döngünün başarılı bir şekilde kullanılmasının en büyük püf noktalarından biri belirlenen koşulların titizlikle ve döngünün başında belirlenen gereksinimlerle doğrulanmasıdır. Yazılım arka arkaya birden fazla döngüyle geliştikçe, yazılım her bir versiyonunun doğrulanabilmesi için belirli testler yapılmalıdır. Testler sürekli tekrarlanarak hata payı düşürülmelidir. Bu model uzun sürebilecek, büyük ve kritik önem taşıyan projeler için daha uygundur. Modelin avantajları; bazı çalışma biçimleri döngünün başında geliştirilir, sonuçlar erken elde edilir, ilerlemeler ölçülür, gereksinim değiştirmek daha az maliyetlidir, küçük parçalara ayrılarak gerçekleştirildiği için hata ayıklamak daha kolaydır, riskler döngünün yinelenmesi sırasında tespit edilir ve bu sürede çözülebilir, risk yönetimini kolaylıkla yapar. Modelin dezavantajları; sistemin mimarisi veya tasarımı açısından sorunlar çıkabilir, küçük projeler için uygun değildir, projenin sonu bilinemeyebilir bu yüzden risk oluşturur, risk analizi yapabilmek için yüksek özelliklere sahip kaynaklara ihtiyaç vardır.

* 1. Prototipleme (Prototyping):

Doğrusal bir modelin döngüsel bir biçimidir. Bu modelde ilk önce gereksinimler hızla toplanır. Yazılımın üreticileri ve hedef kullanıcılar bir araya gelerek elde edilecek çıktılara, bu çıktılar için hangi adımların gerektiğine, ürünün güvenliğinin nasıl sağlanacağına karar verilir. Karar aşamasından sonra üreticiler tarafından hızlı bir tasarım yapılır ve kullanıcının kullanımına ilk örnek sunulur. Kullanıcının değerlendirmesi beklenir, değerlendirmeye göre ilk örneğin üzerinde değişikliklere gidilir. Değişiklik sonrası kullanıcı yeni örneği değerlendirir. Bu süreçte kullanıcının istediği yazılıma yaklaşılmış olur, yazılım ortaya çıktığında ne yapacağına, ne işe yarayacağına dair kullanıcıyla ortak bir noktada anlaşma yapılır. Gereksinim analizinin yapılmasından sonra prototipleme için tasarım yapılır ve böylece geliştirme süreci başlatılır. Sınırlı yineleme sonucunda müşteriye ürünün son hali teslim edilir. Bu modelin çalışmasının en önemli noktası müşteri ve üretici arasındaki iletişimin doğru ve sık olmasıdır. Müşteri ve üretici arasında ne kadar çok iletişim olursa ürün için geri bildirimin artması, istenilene daha çok yaklaşılması beklenir. Prototipleme modelinin avantajları; kullanıcı üretimin her aşamasını ve gereksinimleri görebilir, kullanıcı ve üretici arasındaki iletişim sayesinde karmaşa ve yanlış anlamalar engellenir, risk kontrolü sağlanır, hatalar erken test edileceğinden zaman ve maliyet açısından yarar sağlanır. Dezavantajları; belgelendirme yapılmaz, düzeltme aşaması atlanırsa bu durum risk oluşturabilir, prototip için belirlenen harcamalar düzgün bir yol izlemezse prototip sayısı fazla olabilir.

* 1. Kodla ve Düzelt Modeli (Code & Fix Model):

En basit ürün geliştirme (Cowboy Coding) olarak adlandırılan bir modeldir. Hemen ana problem üzerine odaklanarak sonuca gitme hedeflenir. Test edilmesi gereken ürünün tamamlanmasına odaklanır, bir test ekibi hataları bulmakla uğraşır, üretici kısım da hataları düzeltir. Sonuç ve zaman belirsizliğinden dolayı kaliteli, güvenilir ürün çıkartma olasılığı oldukça azdır. Bir konu araştırılıyorsa ve yeni öğreniliyorsa veya bir problem çözülmeye çalışılıyorsa bu model kullanılabilir.

* 1. Büyük Patlama (Big Bang) Modeli:

Basit bir model olarak bilinen bu model belirli bir süreç gerektirmez, küçük bir planlama ile yürütülebilir. Zor süreçler gerektirmez. Gereksinimler çok fazla analiz edilmez, gerekli herhangi bir değişiklik tüm yazılımı yenileme sürecine götürebilir. Akademik ve küçük projeler için uygun bir modeldir. Nesne yönelimli, karmaşık ve uzun vadeli yazılımlar için riskli ve zayıf bir modeldir. Gereksinimlerin iyi anlaşılmadığı ve son çıkış tarihinin verilmediği ürün için ideal bir modeldir.

* 1. Çevik Modeller (Agile Models):

Eksiksiz çalışan yazılım sürümü sunmayı hedefleyen bir modeldir. Uygulamada çalışan herhangi bir bölümün hızlı bir teslimatı ve müşteri memnuniyeti odak noktasıdır. Prototip oluşturarak ve bu prototipleri kullanıcının gereksinimlerine göre ayarlayarak, hedef sistemi aşamalı bir süreçten geçirerek geliştirmeye çalışılır. Önceki sürümler üzerinden tekrar geçilerek sürümler genişletilir ve sürümün yeni haliyle birleştirilir. Çevik modeller sürekli geri bildirimi vurgular. Bu model belgeleme ve test sürecine daha az önem verir, bu yüzden hızlı gelişme sağlar ancak sorun tespit etme sürecinin uzamasına ve karmaşık hale gelmesine sebep olur. Çevik modeller değişimle başa çıkmada esnek ve başarılıdır. Bu modelde ekip etkileşimi vurgulanır. Proje sık teslim edilir bu da projeyi sürekli doğrular, riski azaltır. Projenin yapım aşamasında gereksinim değiştikçe proje genelinde ciddi değişikliklere gitmek gerekebilir. Bu da uzun zaman dilimlerine tekabül eder. Tamamen müşteri ve geliştirici arasındaki etkileşime bağlı olan bu modelde, müşteri kararsızsa ve net değilse gelişme süreci yanlış yönlendirilebilir, hatalara yol açılabilir. Erken geri bildirim gerektiren başlangıç girişimleri, küçük fonksiyonel parçalara bölünmesi kolay olan projelerde kullanılması uygundur. Prensipleri; müşteriye hızlı ve sorunsuz teslim yapmak, gereksinim değişiklerini kabul etmek ve esnek olmak, kaliteli yazılım teslimatı yapmak, ekibi sürekli iletişim halinde tutmak.

Çevik modellerin avantajları; hızlıdır, değişime açık ve esnektir, bir takım işidir, takım çalışanlarında motivasyon yüksektir bu da verim artışını sağlar. Dezavantajları; kurumsal yapışa uygulanması oldukça zordur, sürekli değişen gereksinimlerden dolayı proje üzerinde aşırı çalışma gerektirir.

Çevik metodojilerinden en yaygın olanı Extreme Programming (XP)’dir. Bu metodolojide müşteri ve gereksinimleri merkezi bir rol alır. XP müşterinin gereksinimlerine çabuk adapte olabilir. Bu metodoloji grup içi iletişime oldukça önem verir. 4 temel değeri vardır. Birincisi basitliktir. Bu temel değer basit yöntemlerle sonuca ulaşmak ister. Bakımı ve geliştirilmesi çok kolaydır. Sadece hızlı ve düşük maliyetli projeler için gerçekleştirilir. İkincisi geri dönüştür. Kalite kontrolü bu değer ile sağlanır. Yazılan programdan geri dönüş alınarak kalite sağlanır, hatalar kaldırılır. Yapım aşamasında sürekli projenin son durumu hakkında geri dönüş sağlar. Üçüncüsü iletişimdir. Çalışanlar sürekli iletişim kurmalıdır, yüz yüze görüşmek büyük önem taşır. Bu temel değer ile yanlış anlamalar ve bilinmezlikler ortadan kalkar. Dördünce ve en sonuncusu cesarettir. Geliştiriciler programı yaparken başarısızlıktan korkmamalıdır, yaptıkları projeden memnun kalmadılarsa çalışmayı çöpe atmadan korkmadan, yeniden başlanabilmelidir. Diğer bir metodoji Scrum’dır. Scrum bütünü parçalayarak, tekrara dayanan bir yöntem izler. Düzenli planlamayla hedefe ulaşmaya çalışır. İhtiyaca yönelik bir yapısı vardır. Gereksinimler değiştiğinde kolayca değiştirilebilir, esnektir. Müşteri ihtiyacına göre şekillenir, müşterinin geri bildirimine göre yapılandırma sağlar. Gereksinimlerin düzgün tanımlanmadığı projeler için uygundur. 3 prensibi vardır: Birincisi şeffaflıktır. Proje ilerleyişi, gelişmeler ve proje esnasında meydana gelen sorunlar herkes tarafından görülebilir olmalıdır. İkincisi denetlemedir. İlerleyiş sürekli ve düzenli bir biçimde kontrol edilir. Üçüncüsü ise uyarlamadır. Proje yapılan değişikliklere uyum sağlayabilmelidir. 3 adet Scrum kavramı vardır. Product Backlog, proje gereksinim listeleridir. Müşteriden gereksinimler alınır ve ihtiyaca göre ekleme veya çıkarma yapılabilir. Bu sayede projenin her aşamasında değişim meydana gelebilir. Sprint Backlog, Product Backlog’da çıkarılan gereksinim listesi detaylı olarak zaman çizelgesi çıkartılır. Scrum Daily Meeting, her gün geliştirme ekibiyle Scrum Daily Meeting yapılır ve projenin durumu hakkında bilgi alınır. Scrum rolleri 3 adettir: Product Owner (Ürün Sahibi), müşteri ve geliştirme ekibi arasındaki iletişimi sağlar. Scrum Master (Scrum Yöneticisi), Scrum kurallarını, teorilerini ve basit yöntemlerini iyi bilir ve geliştirme ekibinin bu kurallara uymasından sorumlu olan kişidir. Ekibin çalışmasını engelleyecek herhangi bir durumu ortadan kaldırır. Scrum Team (Scrum Takımı), devamlı birbirleriyle iletişim halinde olan ve hedefe ulaşmaya çalışan kişilerden oluşur. Scrum toplantıları 3 adettir: Sprint Planning (Sprint (Koşu) Planlama), gereksinimleri geliştirme ekibi küçük görevlere ayırır ve ekipte bulunan her bir kişi bu görevlerden kendisine en uygun olanı seçer. Risk değerlendirilmesi yapılır ve proje tekrar gözden geçirilerek gereksinim değişiklikleri belirlenir. Daily Scrum Meeting (Günlük Scrum Toplantısı), her gün belirlenen işe başlamadan önce yapılan 15 dakikalık toplantılardır. Dün ne yaptım? Bugün ne yapacağım? Beni ne engelliyor? Sorularına cevap verir. Belirlenen görevin gün içerisinde bitmesi imkansızsa görev tekrar parçalanır ve gerçekleştirme ekibi arasında dağıtılır. Sprint Review (Sprint Değerlendirmesi), her sprint sonunda yapılır. Ortaya çıkan ürün değerlendirilir ve müşterinin gereksinimlerine uygun olup olmadığı kontrol edilir. Eğer herhangi bir hata fark edilirse hata düzeltilir.

Scrum bileşenleri: Ürün Gereksinim Dokümanı (Product Backlog), ürün gereksinim listesi basit bir listedir ve ürün gereksinim dokümanı geçerli ve kullanılmaya devam edebilmesi için canlı olması gerekir. Gereksinim listesi genellikle toparlanan kullanıcı hikâyelerinden oluşmaktadır ve ürün sahibi tarafından önceliklerine göre düzenlenir. Sprint (Koşu) Dokümanı (Sprint Backlog), halledilecek işleri ve görevleri gösterir. Ekipteki elemanların günlük hangi görev üzerinde çalıştığı ve görevi bitirip bitirmediği hakkında bilgi verir. Bitmeyen görevler ise kırmızı bir nokta ile işaretlenerek engellerin kolayca tespit edilmesi sağlanır. Sprint Kalan Zaman Grafiği (Burndown Chart), geri kalan çalışma bu grafik ile görselleştirilir. Belirlenen görevlerin ne kadarının yapıldığı ve normalde ne kadar yapılması gerektiğini karşılaştırabilmeyi sağlar. Grafik sayesinde proje bitimine ne kadar süre kaldığı belirlenir ve müşteriye bildirilir.

1. SCRUM GÜNÜMÜZDE NEDEN POPÜLER?

Scrum günümüzde kullanımı oldukça artan, trend haline gelen bir yazılım geliştirme anlayışıdır. Bu anlayışın günümüzde trend haline gelmesinin en büyük nedenleri arasında tekrara dayalı, gözleme dayalı ve detaylı olması bulunur. Yapılacak projeyi küçük parçalara ayırarak her bir detayını tekrar tekrar analiz edip incelemesi ürünün kalitesini arttırdığından dolayı çoğu geliştiriciler tarafından proje geliştirilirken Scrum geliştirme anlayışından faydalanılır.

1. SONUÇ

Yazılım yaşam döngüleri üretim aşamasının her alanında yer alır. Üretilecek ürünün çeşitliliği ve özelliklerine göre geliştiriciler tarafından bir yazılım yaşam döngüsü modeli seçilir ve üretim kısmı bu döngü modeli sayesinde daha kolay hale getirilir. Ürünlerin kalitesi artırılır.

KAYNAKLAR:

Linkedin: <https://www.linkedin.com/in/elif-r%C3%BCmeysa-turan-3bb082235/>

GitHub: <https://github.com/elifrturan>

Medium: <https://medium.com/@elifrturann>

1. J. Glenn Brookshear, Dennis Brylow (2014). Bilgisayar Bilimine Giriş, (Çev.) Dr. Öğr. Üyesi Birim BALCI, Nobel, Ankara.
2. Doç. Dr. Deniz KILINÇ, Yazılım Mühendisliği Temelleri ders notları.
3. <https://caglartelef.com/yazilim-yasam-dongusu/>
4. Deniz KILINÇ, Medium, Yazılım Yaşam Döngüsü Temel Aşamaları.
5. Seçil ÇOR, Medium, SCRUM NEDİR?
6. <https://iskulubu.com/yazilim/yazilim-gelistirme-yasam-dongusu/>
7. <https://medium.com/@ahmetuyar/extreme-programming-xp-nedir-ddc003a515c4>
8. <https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-62915545c51e>