Yazılım yaşam döngüsü yazılımın üretim ve kullanımındaki tüm aşamaları kapsar. İhtiyaçlar zamanla değiştiğinden sürekli bir döngü halinde olmalıdır. Tek yönlü yani doğrusal olarak kesinlikle düşünülmemelidir. Yazılım yaşam döngüsü Risk düşürücü çözümler içerir ve kaliteyi arttırır. Bu metodu kullanmanın birçok avantajı vardır örneğin plan aşamasında, analiz aşamasında veya tasarım aşamasında bir problem çıktığında nerede çıktığı, yeni adımda eklenecek şeylerin hangi adımda eklenmesi gerektiği ortaya çıkarılabilir ve problemi rahat bir şekilde tespit edip ortadan kaldırılır. Avantajları olduğu gibi bazı dezavantajları da vardır örneğin bu metodu kullanmak uzmanlık gerektirir ve küçük projelerde kullanılması pek uygun değildir bu metot daha çok büyük projelerde kullanılır. Yazılım yaşam döngüsü gereksinim, analiz, tasarım, gerçekleştirme ve bakım olmak üzere temel olarak 5 adım içermektedir. Gereksinim aşaması müşterinin istekleri doğrultusunda programdan beklenilenin öğrenildiği aşamadır. Doğru ve eksiksiz yapılması çok önemlidir. Analiz aşaması yapılacak işin gereksinimlerinin ve işlevlerinin incelendiği çıkabilecek sorunların ayrıntılı olarak incelendiği aşamadır. Tasarım aşaması üst seviye mimari tasarım ve detaylı tasarım olmak üzere iki adımda gerçekleşir. İki adımda belirlenen niteliklere uygun olarak yazılım sisteminin tasarımının oluşturulduğu aşamadır. Gerçekleştirme aşaması Tasarım aşamasında kurulan temellere bağlı kalınarak yazılımın yazıldığı test edildiği aşamadır. Bakım aşaması programı müşteriye teslim ettikten sonra zaman içinde çıkan problemlerin çözülmesi ve ayrıca yeni gereksinimlere ihtiyaç duyulursa programın güncellenmesini kapsayan aşamadır. Yazılım geliştirmede geçmişten beri birçok model kullanılmıştır. Projeye en uygun modeli seçmek çok önemlidir bu sayede hem zamandan hem de maliyetten tasarruf yapılır. Bu yazımızda en temel modelleri açıklayacağız.

**1.Gelişigüzel Model:** 60’lı yıllarda ortaya çıkan bir modeldir. Genellikle tek kişinin yaptığı küçük projelerde kullanılır. Yazılım geliştirme ortamında herhangi bir model veya yöntemin kullanılmadığı zamanlarda kullanılmıştır. İzlenile bilirliği ve dolayısıyla bakım yapılabilirliği oldukça zordur bazı durumlarda olanaksızdır. **2.Barok Modeli:** 1970’li yıllara kullanılan artık günümüzde kullanılmayan bir modeldir. Barok modeli, planlama, çözümleme, tasarım ve gerçekleştirim işlevleri içermektedir. Bu modelin ayırıcı özelliği, belgelemeye ayrı bir önem vermesi ve belgelemeyi bir süreç olarak ele almasıdır. Barok modeli gerçekleştirim döngüsüne ağırlık verir. Kullanılmamasının temel nedenlerinden biri aşamalar arasında gereken geri dönüşlerin tanımlı olmamasından kaynaklıdır. **3.Çağlayan Yaşam-Döngü Modeli:** Geçmişte en popüler yazılım geliştirme metodudur.1970-1980 yıllarında geliştirilen statik bir yazılım yaşam döngüsü modelidir. Çağlayan modelinde aşamaların en az bir kere tekrarlanmasıyla gerçekleştirilir ve 5 aşamadan oluşur. Bu aşamalar planlama, analiz, tasarım, uygulama ve testlerdir. Barok Modelinden farklı olarak dokümantasyon ayrı bir aşama olarak alınmamıştır ve aşamalar arasında geri dönüşlerin tanımlıdır. Ancak, çağlayan yaşam-döngü modelinin kullanımında dikkat edilmesi gereken önemli hususlar vardır. Bunlardan en önemlisi, her ne kadar model içerisinde aşamalar arasında geri dönüşler yapılabilse de analiz aşamasında mümkün olan tüm detayın tasarıma yansıtılabilmesi için müşteri ve sistem gereksinimlerinin en ince ayrıntısına kadar belirlenmesi gerekir. Aksi taktirde yeni istekler ortaya çıktığında tekrardan ilk aşamaya dönülmesi gerekmektedir dolayısıyla maliyetler çok yükselir. Bundan dolayı bu modeli kullanan ekipler analiz ve tasarım aşamasında çok vakit harcamaktadırlar. Kendini yenilemeyen güncellenme istemeyen yazılım projeleri için uygun bir modeldir ancak yazılım dünyasında çoğu yazılım projesi zaman geçtikçe kendini yenileme ihtiyacı duyar, yeni istekler ortaya çıkar. Bu yüzden bu model artık günümüzde sıkça kullanılmaz. Şelale modeli yazılım projeleri için en kötü yaklaşımlardan birisidir denilebilir. Yine de ilk olması ve basit olması ve diğer yaklaşımlara temel teşkil etmesi açısından önemlidir. **4.Fıskiye Modeli:** Bu model Henderson - Sellers ve Edwards tarafından oluşturulmuştur. Çağlayan modelinden esinlenerek oluşturulmuş bir modeldir. Çağlayan modelinde farkı her aşamada geriye dönük döngüler bulunmaktadır. **5. V-Şeklinde Model:** iki aşamalı olarak ihale edilmesi için oldukça uygundur. İlk ihalede kullanıcı modeli hedeflenerek, iş analizi ve kabul sınamalarının tanımları yapılmakta, İkinci ihalede ise ilkinde elde edilmiş olan kullanıcı modeli tasarlanıp, gerçekleşmektedir. V-şeklinde model 9 aşamadan oluşur bu aşamalar planlama, ihtiyaç, üst seviyeli tasarım, detaylı tasarım, kodlama, birim testleri, entegrasyon testleri, kabul testleri ve bakımdır. Üst seviye tasarımda daha genel bir tasarım yapılır detaylı tasarımda ise önceki aşamada yapılan tasarımın detaylarına inilir. Kodlama aşamasına kadar çağlayan modeline benzer. Buradan sonra birim testleri yapılır bu aşamada her üretilen modülün testi yapılır. Entegrasyon testinde modüllerin birbiriyle çalışma durumu kontrol edilir. Kabul testlerinde müşteride teste dahildir. Uygulama müşteri tarafından test edilir ve son olarak bakım aşaması bulunur. Çağlayan modelinden farklı olarak ise v süreç modelinin her aşamasının karşılık bulduğu farklı aşamaların olmasıdır bu sayede basamakların doğruluğu karşılığı olan aşamaya bakılarak daha rahat yapılabilir. Detaylı tasarım aşamasına karşılık olarak birim testleri, üst seviye tasarıma karşılık entegrasyon testleri vardır, İhtiyaç analizlerine karşılık kabul testleri ve son olarak planlamanın da daha sonra bakım aşamasında karşılığı vardır. **6.Prototipleme Modeli:** Bu modelinin temel amacı prototip çıkarmaktır. Bu modelde hızlı plan, hızlı tasarım, hızlı kodlama ön plandadır. Bu modelde izlenen adımlar belirsizliği tanımlamak çözümleri tanımlamak prototip çalışması yapmak belirsizliğin sonucunu elde etmektir. Modelde izlenen en temel 4 adım belirsizliği tanımla, çözümleri tanımla, prototip çalışması yap, belirsizliğin sonucunu elde et adımlarıdır. Prototip modelinin zayıf kaldığı yönü kaynak maliyetini planlayamamasıdır. **7.Arttırımlı Model:** Bu modelde bir döngü söz konusudur.Planlama aşaması başlanır ve ardından bir döngü içine girer. Bu döngü değerlendirme, istek analizleri, analiz ve tasarım, uygulama ve test aşamalarını içerir. Bu döngü tamamlandıktan sonra canlıya geçiş aşamasına sıra gelir buradan bir prototip çıkarılır ve testler yapılarak bu döngü devam eder. Problemler çıktıkça test aşamasında problemler ortadan kaldırılır ve problemler sistemi besler. Böylelikle her döngü bir tur attığında arttırılmış bir prototip oluşur. Yeni istekler geldikçe bu döngüye katılır ve süreç devam eder. Bu modelde ürünler üretilirken çekirdek yapı korunarak ürün geliştirilir. Uzun zamanlı projeler için eksikler çıktıkça yama yapmak tipindeki projeler için uygundur. Kullanım ve üretim eş zamanlıdır. **8.Horozonik (Spiral) Model:** Spiral model dört bölümden oluşur ve bu bölümler sürekli bir döngü içerisinde tekrarlanır. Dört aşama hedeflerin belirlenmesi, risklerin belirlenmesi ve çözülmesi, geliştirme ve testler ve son olarak sonraki dönüşün planlanmasından oluşur. Bir döngü içerisinde ilk olarak isterler belirlenir, operasyon hakkında fikir sahibi olunur, ihtiyaçlar belirlenir, daha sonra risk analizleri yapılıp prototip çıkartılır. Böylece döngü tamamlanmış olur. Bu döngü projenin büyüklüğüne göre belirlenen sayıda döner. Döngünün bitiminde çağlayan modeline benzer bir yol izlenerek detaylı tasarım, kodlama, entegrasyonlar, testler ve uygulama aşamalarından geçerek hedefe ulaşılır. Bu modelde risk analizi önemli bir yere sahiptir. Her döngü bittiğinde bir prototip oluşturulur ve döngü devam ettikçe bu prototipler gelişir. Prototiplerin gelişmesi kullanıcının projeyi daha gelişme aşamasındayken görmesine olanak sağlar. Projenin büyüklüğüne göre daha fazla döngü dönebilir bu durum maliyeti arttırır. Küçük ve düşük riskli projeler için oldukça pahalı bir sistemdir bu yüzden büyük sistemlerde kullanılması daha uygundur. **9.Evrimsel Geliştirme Süreç Modeli:** Bu model İlk tam ölçekli olan modeldir. Daha çok coğrafik olarak geniş alana yayılmış, çok birimli organizasyonlar için önerilmektedir. İlk olarak pilot uygulama üretilir ve kullanılarak bazı birimlerde test edilir. Pilot uygulamanın problemleri ve eksikleri test aşamasında öğrenilerek pilot uygulama güncellenir. En sonunda son sürüm tüm birimlere dağıtılır. **10.Araştırma Tabanlı Model:** Geliştirilen yazılımlar genellikle sınırlı sayıda kullanılır ve kullanım bittikten sonra işe yaramaz hale gelir ve atılır bu yüzden bu modele yap-at prototipi de denir. Bu modelin araştırma ortamları tümüyle belirsizlik üzerinedir ve bir sonraki adımın iş tanımları büyük ölçüde bir önceki adımın sonuçlarına bağlıdır. Bundan dolayı bu modelde zaman ve fiyat öngörülmesi oldukça zordur sabit fiyat sözleşmesi yapılamaz. Modelde yapılan işlerden elde edilecek sonuçlar belirgin değildir. **11.Kodla ve Düzelt:** Küçük projelerde ya da kısa ömürlü prototiplerde uygulanır. Genellikle öğrenci projelerinde dokümantasyon yapılmadan kullanılır. Eğer kodlama sırasında bir hata yapıldığında yada güncelleme gerektiğinde kodlar sonradan değiştirmek için planlanmadığından esnek değildir, değiştirilmesi zordur. Yazılım geliştirmenin en kolay yollarından biridir ancak değiştirilmesi zor olduğundan en pahalı modeldir.

**Çevik Yazılım Geliştirme:** Yazılım geliştirme süreci uzun ve yaşayan bir süreçtir. Sürekli kendini yinelemesi ve geliştirmesi gerekmektedir. Yazılım sürecinde yapılan hataların geç fark edilmesi, sürekli kendini güncelleme istemesi ancak bu güncelleme aşamasının uzun sürmesi ve dolayısıyla yazılımın zamanında ortaya çıkmaması gibi sorunlar ortaya çıkmaktaydı. Bu sorunları ortadan kaldırmak amacıyla 1990’lı yılların sonlarında çevik yazılım geliştirme metotları geliştirilmiştir. Çevik yazılım geliştirme metotlarına 1970’li yıllarda da rastlansa da kullanımı 1990’lı yılların sonunda hız kazanmıştır. Bu metotların kullanımı tüm dünyada kısa bir zaman içerisinde kendini kanıtlayarak popülaritesi artmıştır. Şu anda dünyada irili ufaklı birçok şirket tarafından yoğun bir şekilde kullanılmaktadırlar. Çevik yaklaşımlar uygulandığında proje üretkenliği, projenin değişimlere açık olması, kalitesi ve pazara çıkış hızı artmaktadır. Bunların doğrultusunda projelerin başarı oranları yükselmiştir. Bu metotların kullanıldığı projelerin bitirilme hızı, proje ekibinde çalışan tüm ekip üyelerinin sürekli iletişim halinde olmasına bağlıdır. Bu metotta proje 2-4 haftalık süreçlere ayrılarak tamamlanır ve bu sayede yapılabilecek bir hatanın geriye dönülerek oldukça rahat ve az maliyetle düzeltilebilir. 2001 yılında, dünyanın önde gelen çevik yazılım geliştiricileri ortak bir zeminde buluşabilmek adına bir araya gelerek çevik yazılım geliştirme manifestosunu ve çevik yazılımın prensiplerini yayınlamışlardır. Böylelikle çevik metotların projelere genel bakış açıları ifade edilmiştir. Bu manifestoda süreçler ve araçlar yerine bireyler ve etkileşimler, kapsamlı belgeler yerine çalışan yazılım, sözleşme görüşmeleri yerine müşteri ilişkileri, plan izleme yerine değişikliğe açıklığın daha önemli ve öncelikli olduğu belirtilmiştir. Bu prensiplerle hızlı ve sürekli kullanılabilir, değişikliklere açık ve kaliteli bir yazılımı amaçlanmaktadır. Yazılım sürecinde birbiriyle iletişim halinde ve sürekli müşteriyle de iletişim halinde olan bir ekip olmalıdır. Prensiplere göre motivasyonu yüksek ve güvenilir bir ekip, bu ekip içerisinde kaliteli ve yüz yüze bir bilgi akışı olmalıdır. Prensiplere bağlı kalınırsa kısa sürede müşteri memnuniyeti sağlanır, üretkenlik ve yazılım kalitesi artar, maliyet düşer ve yazılım projelerinin başarısı oldukça yüksek olur. Yazımızda en çok uygulanan çevik yazılım metotlarına değineceğiz.

**1.Extreme Programming (XP):** En popüler çevik süreçlerden biri XP’DİR. Kent Beck ve arkadaşları tarafından 1996 yılında bir proje bünyesinde oluşturulmuştur. XP sürecinde müşteri temsilcisi ve gereksinimlerinin büyük önemi vardır. Takım ve müşteri temsilcisi sürekli iletişim halindedir. Bu iş temsilcisi diğer müşteri gereksinimleri otaya koyar. Müşteri olayı ne kadar iyi bilirse o kadar iyi olur. Tüm takım bir grup halinde yada ikişerli gruplar halinde çalışırlar. Basitten karmaşığa doğru ilerleyen bir yapıya sahiptir. Extreme programlamada ortada 12 temel pratik konulmuştur. 1.1.Planlama Oyunu: Ekip ve müşterinin olduğu bir toplantıda o işin ne kadar sürede yapılacağının öngörülmesi ve o süreye bağlı kalınarak çalışmaların sürdürülmesidir. 1.2.Ekipte Müşteri: Müşteri ve ekip aynı ortamda bulunmaktadır bu sayede müşteri her adıma dahil olur ve isteklerini dile getirir. 1.3.Önce Test: Kod yazılmadan önce test programları hazırlanır bu sayede oluşan hatalar erkenden ortaya çıkarılarak daha güvenli bir yazılım oluşturulur. 1.4.Basit Tasarım: Müşterinin ihtiyaçlarını karşılayacak en basit tasarım çıkarılır detaya inilmez. 1.5.Çiftli Programlama: Programcılar tüm kodları çiftler halinde yazarlar, tüm işleri hep beraber yaparlar. Dolayısıyla takımın her üyesi her kodu anlama ve geliştirme olanağına sahiptir. Usta çırak ilişkisi vardır ve olay farklı bakış açılarından görülür. 1.6.Sürekli Entegrasyon: Her kodlama görevi tamamlandıktan sonra ana sistemle bütünleştirilmelidir. Bu sayede oluşabilecek hatalar erken tespit edilir. 1.7.Kısa Aralıklı Testler: Proje kısa aralıklara bölünür ve her bir parça teslim tarihinde müşteriye teslim edilir. Bu sayede müşteri projenin ilerleyişini takip edebilir. 1.8.Yeniden Yapılandırma: Kod tekrarlarından uzak durmak, sade ve temiz kod elde etmek ve müşterinin isteklerini mevcut duruma bakarak geliştirmek için sürekli olarak yeniden yazılmalıdır. 1.9.Ortak Kod Sahiplenme: Geliştirilen yazılım kodu bütün ekibin ortak malı olarak görülür. Böylece bir ekip üyesi başkasının yazdığı kodlara erişerek kodu daha iyiye götürebilir. 1.10.Benzetim: Yazılımda sistem küçük parçalara bölünür ve her bir parça başka bir sistemle benzeştirilir. 1.11.Kodlama Standardı: Bütün kodu yazmada koca bir ekip olduğundan kodun anlaşılabilmesi için önceden kodlama standartları tanımlanır. 1.12.Haftada 40 Saat: XP yaklaşımında, sürekli şekilde fazla mesai yapma bulunmamaktadır. Planların ve öngörülerin haftada 40 çalışma saati üzerinden yapılmasını önermektedir. Dağınık şekilde 60 saat çalışma yerine tamamen konsantre olunmuş verimli 40 saat tercih edilmesi gerektiği vurgulanmaktadır. Fazla mesaide verim artışının olmadığı hatta yapılan hatanın arttığı düşünülmektedir.

**2.Scrum:** Günümüzde en çok tercih edilen çevik yazılım geliştirme metodudur. Çevik yazılım geliştirme yöntemlerini kullanan şirketlerin %70’e yakın bir kesimi scrum kullanır. Günümüz de bir metotta aranan en önemli özellik değişikliklere hızlı adapte olabilmesidir Scrum, değişen gereksinimlere hızlı uyum sağlamayı mümkün kılan, yazılım hatalarını azaltan, yazılım ekiplerinin entegrasyonuna zemin hazırlayan bir yapısı olduğundan bu kadar popüler olarak kullanılmaktadır. Scrum’ın kelime anlamı Amerikan futbolundan gelmektedir. Tüm ekip bir araya gelir ve top kazanmak için tüm enerjilerini harcarlar buna scrum denir. Scrum, ilk bakışta çok basit kuralları olan bir yönetimsel modeldir. Gereksinimleri açıkça belirli olmayan, değişime açık, karmaşık yazılım projelerinin yönetimi için uygulanmaktadır. Yenilemeli bir yazılım geliştirme modelidir. Scrum metodolojisi, oldukça esnek bir geliştirme süreci sürdürebilmek için tasarlanmıştır. Scrum, ürün çıktısını planlamayı ve proje ilerlerken değişkenleri yönetebilmeyi mümkün kılmaktadır. 1-2 hafta süren sprint olarak adlandırılan hızlı döngü aşamaları vardır. Sprintin sonunda tüm ekip toplanır yapılan iş gözden geçirilir, sonraki adımlar planlanır ve daha sonrasında yeni sprint başlar. Scrum metodolojisinde takım çalışmasının oldukça önemli bir yeri vardır. Bütün takım devamlı iletişim alinde olmalıdır. Scrum da üç adet özel rol vardır. Bunlar: 1.Ürün Sahibi: Yapılması gereken tüm işler için öncelik belirler, iş ve pazar gereksinimlerine odaklanır. 2.Scrum master: Takımın mümkün olan en iyi işi yapmasına yardımcı olur. Görevleri arasında toplantıları düzenlemek, sprintleri denetlemek, yapılan ve yapılacak işleri denetlemek vardır. Aynı zamanda ekipte bulunmayan yönetici ve müşteriyle de iletişim içerisinde olmak orundadırlar. 3.Scrum Takımı: Scrum Takımı beş ila yedi üyeden oluşur. Projedeki herkes birlikte çalışır, birbirlerine yardım eder. Scrum Ekibi her sprint için bir plana sahiptir; Her iterasyonda ne kadar iş yapabileceklerini tüm ekip birlikte tahmin ederler. Scrum da uygulanan üç çeşit temel toplantı vardır. Sprint planlama, sprint gözden geçirme ve günlük scrum toplantısı. Sprint planlama toplantısında sprint süresince yapılacak işler planlanır. Scrum takımıyla gerçekleştirilir ve maksimum 8 saat süren bir toplantıdır. Sprint gözden geçirme toplantısında sprint süresince neler yapıldığı konuşulur. Ortalama 4 saat olan bu toplantının amacı bir önceki sprint i.in geri bildirim almak ve iş birliğini arttırmaktır. Günlük scrum toplantısı 15 dakika süren küçük toplantılardır. Hangi işlerin yapıldığı, bir sonraki toplantıya kadar hangi işlerin tamamlanmasının planlanır. Scrumın rolleri, eserleri, etkinlikleri ve kuralları değiştirilemez. Scrumın bazı kısımlarını uygulamak her ne kadar mümkün olsa da sonuç Scrum olmaz. Scrum sadece bir bütün olarak vardır ve diğer teknikler, metodolojiler ve pratikler için iyi bir hazne görevi görür. Dünyadaki en büyük (Google, Microsoft, YAHOO!, IBM, IKEA, SIEMENS, NOKIA, BBC…) şirketler scrum metodolojisini kullanmaktadır.

Sonuç olarak makalemizde bir projenin üstesinden gelinmesi için sadece kodlamanın yeterli olmadığına, projemizin özelliklerine en uygun olan metodu seçip projemizin tüm aşamalarında bu metoda bağlı kalınarak yapılması gerektiğine değindik. Bu sayede projemizi kaliteli, hızlı ve az maliyetle bitirebiliriz.

Umut Akter

180601026