YAZILIM YAŞAM DÖNGÜ MODELLERİ

Bir yazılım üretim ve kullanımda geçirdiği aşamalar yazılım yaşam döngü modellerini oluşturur. Yazılım yaşam döngüsü yazılımları iyileştirmek, geliştirmek, test etmek ve sürdürmek için oluşturulan süreçlerdir. Yazılımın işlevi ve ihtiyaçları sürekli değişken ve gelişmek üzerine olduğundan lineer bir yol yerine döngü şeklinde düşünülmelidir. Bir yazılımda planlı ilerlemek, her aşamada geri dönülebilir ve test edilebilir olması gerekebildiğinden yazılım yaşam döngü modellerini kullanmak gerekir.

Yazılım yaşam döngü modelleri belirli temel adımlardan oluşur. Yaklaşık beş adım vardır. Bu adımlar; gereksinim (requirements), analiz/çözümleme (analysis), tasarım (design), gerçekleştirme (implementation), bakım (maintenance)dır. Gereksinim aşamasında projenin planlaması ve fizibilite çalışması (maliyet ve olabilirlik açısından araştırma ve bilgi edinme) yapılır. Analiz aşamasında projenin tamamlanacağı süre, risk durumları, işlevleri ve gereksinimleri analistler ve mühendislerin oluşturduğu grup tarafından ayrıntılı olarak belirlenir ve bu bilgiler belirli bir formatta dokümantasyon haline getirilir. Tasarım aşamasında nasıl ve ne sorularına cevap aranır ve tasarlama kısmı başlar. İki tasarım tekniği vardır bunlar soyutlama ve modellemedir. Soyutlama (abstraction) tekniği bazı durumlar ve olayların problemin çözümünü kolaylaştırmak için göz ardı edilmesidir. Bu şekilde önemli ve istenen yerlere odaklanma sağlanır. Modelleme tekniğinde ise diyagram ve modeller kullanılır. Ayrıca iki tasarım yapısı vardır. Bunlar detaylı tasarım ve üst düzey ve mimarı tasarımdır. Detaylı tasarımda yazılımın bileşenleri ve ayrıntıları yer alır. Üst düzey ve mimarı tasarımda ise sistemin genel yapısı yer alır. Gerçekleştirme aşaması kodlama ve testi içinde barındırır. Kodlama kısmında ürün programlanır ve birleştirilir. Test kısmında sorunların, hataların ve eksiklerin belirlenmesi için test yapılır. Sağlıklı bir yaşam döngüsü için test aşaması analiz aşamasından beri yapılmalıdır. Bakım aşaması ürünün tesliminden sonra çıkar. Bu aşamada teslim edilen ürüne varsa yeni gereksinimler eklenir veya varsa hata giderme yapılır. Geri bildirilere göre güncelleştirmeler yapılabilir.

Farklı projeler ve farklı dinamikler için farklı yazılım yaşam döngü modelleri vardır. Bunlar; Gelişigüzel Model, Barok Modeli, Çağlayan Yaşam Döngü Modeli, V Süreç Modeli, Helezonik (Spiral) Model, Artımsal Geliştirme Süreç Modeli, Kodla ve Düzelt Yaşam-Döngü Modeli ve Çevik (Agile) Model’dir.

Gelişigüzel Model; 1960’larda kullanılmıştır. Aslında bu ortamda bir model veya yöntem yoktur. Genellikle bir kişinin üretim yaptığı bu yöntemde geliştirme, değiştirme yapmak; test ve bakım aşamalarını uygulamak oldukça zordur. Basit bir programlama içerir.

Barok Modeli; 1970’lerde kullanılmıştır. Bu model lineer bir şekilde ilerler ve aşamalar arasındaki geri dönüşlerin nasıl olabileceğini söylemez. Belgeleme ayrı bir aşama olarak ele alınır, geliştirme ve test aşamalarından sonra gelir. Günümüzde bu yazılım modeli kullanılmamaktadır.

Çağlayan Yaşam Döngü Modeli; geçmişte zamanının en popüler yazılım geliştirme modelidir. Geleneksel yazılım geliştirme modeli de denilebilir. Bu modelde her aşama tamamen bitirilerek diğerine geçilebilir. Bu her adımlarda da dokümantasyon yazılmasını söyler. Fakat Barok Modelinden farklı olarak dokümantasyonu ayrı bir aşama olarak değil, her aşamada gerçekleştirilen modelin bir parçası gibi görür. Her aşamada dokümantasyon ve test yapılır. Aşamalar arasında geri dönüşler olabilir. Analiz ve tasarım aşamalarının çok detaylı çalışma istemesinden dolayı bu aşamalar çok zaman alır. Çok fazla emek ve süre gerektirir. Gereksinimlerin değişmesi, güncellenmesi, hata ve eksiklerin giderilmesi ile olabilecek değişikliklerin maliyeti yüksek olur ve müşteriye ulaşması zaman alır. Bu sürece müşteri dahil edilmediğinden de yazılım tamamlanınca geri dönüş yapabilir ve bu da maliyeti ciddi oranda arttırır. Bu model ile yapılan programlarda yazılım ekibini mutsuz olmakta ve işin planlama, tasarım, analiz vs. kısmına yeterli önem verilmemektedir. Bu yüzden bu model çok iyi tanımlanmış ve yapılması az zaman gerektiren projelerde kullanılır.

V Süreç Modeli; V şekline şemaya sahiptir ve V şeklinde ilerler. V şeklinin sol tarafı üretim ve sağ tarafı sınama(test) işlevlerini ilgilendirir. Sağ tarafta yapılan test kısmında hata bulunursa yatay olarak karşılık gelen sol taraftaki üretim işlevini ilgilendirir. V Süreç Modelinin üç temel çıktısı vardır: Kullanıcı Modeli, Mimari Model ve Gerçekleştirim Modeli. Kullanıcı Modeli, kullanıcı ile olan ilişkileri içerir ve test belirtmeleri yapılır. Mimari Model, sistemin tasarımı ve tüm sistemin testine ilişkin işlevleri bulundurur. Gerçekleştirim Modeli, yazılımın kodlanması ve testine ilişkin işlevleri bulundurur.

Helezonik(Spiral) Model; ara ürün üretimi ve bu ara ürünlerin kullanıcı tarafından sınanmasına dayanır. Planlama, risk analizi, üretim, kullanıcı değerlendirmesi aşamalarından oluşur. Planlama; ara ürün üretiminde plan, amaç belirleme ve ara ürünlerin entegrasyonunu içerir. Risk analizi; risk durumlarının araştırılması ve belirlenmesini içerir. Üretim; ara ürünün üretilmesini içerir. Kullanıcı değerlendirmesi, ara ürünün kullanıcı tarafından yapılan sınama ve değerlendirmelerini içerir. Risk analizi ön plandadır. Gereksinim, analiz vs. Araştırmaları yoktur. Döngüleri vardır, her döngü bir faz anlamına gelmektedir. Kullanıcı bu modelde erken katılım sağladığı için ürünün sınanması ve gereksinimlerin anlaşılması istenmeyen durumları eler (kullanıcı katkısı). Yönetici ve yazılımcılar proje boyunca iletişim halinde olduklarından kolay bir akış olur (yönetici bakışı). Yazılımın kodlanması ve testi daha erken başlar (yazılım geliştirici bakışı).

Artımsal Geliştirme Süreç Modelinde; ürünün tam halde teslimi değil, parçalara bölünüp geliştirilmiş teslimi yapılır. Kullanıcı gereksinimleri öncelik sırasında göre belirler ve öncelikli gereksinimler erken teslim edilir. Her yazılım sürümü birbirini kapsar. Uzun projeler ve sistemin parçalarının da çalışabileceği projeler için bu model uygundur. Teslim edilen parça kullanılırken aynı zamanda başka parçası üretilir. Böylelikle işlev giderek artar. Bu model yinelemeli (iteratif) yapıya sahiptir.

Kodla ve Düzelt Yaşam Döngü Modeli; birkaç yüz satırdan oluşan programlar için uygundur. Yazılım direkt gerçekleştirilir ve müşterinin gereksinimleri karşılanana kadar geliştirilir. En pahalı ve en kolay yazılım geliştirme yoludur. Kodlamadan sonra gelişim ve değişiklikler maliyeti arttırır. Dokümantasyon yoktur ve bu yüzden bakımını yapmak çok zordur. Bir süre sonra emeklilik (retirement) safhası vardır. Küçük ve tecrübesiz firmalarda sık rastlanan bir modeldir.

Çevik (Agile) Modeli; 1990’lı yılların sonuna doğru geliştirilmiştir. Modelin amaçları; ürünü çabuk çıkarabilme, ucuz çözümler üretebilme, değişikliklere hızlı davranabilme ve müşteriyle iletişim içinde hızlı ürün sunabilmedir. Bu yazılım modeli yazılım ekibinin motivasyonunu yüksek tuttuğu için verimliliği yüksektir, üretkenlik fazladır ve ürün kalitesi yüksektir. Proje boyutu ne olursa olsun küçük parçalara, yinelemelere (iteration) ayrılır ve her parçaya proje gözüyle bakılır. Küçük proje parçalarında ekip, müşteriye proje ile ilgili bilgi verir, müşteri ile sürekli iletişim içindedir, müşteri ve ekip arasındaki bu iletişim ile proje hızı gelişir. Her yineleme 2-4 hafta kadar sürer ve müşteriye çalışan yazılım teslim edilir. Müşteri iletişim içine olduğu için ve sürekli çalışan yazılım aldığı için memnuniyeti artar. Küçük proje parçalarından oluştuğu için de hata ve eksikliklerin giderilmesi basit ve ucuza olur. Yani bu metotla verimliliği, hızı, motivasyonu yüksek bir ekip; iletişim halinde, memnun müşteri; esnek, hızlı, hata oranı haz, ucuz yazılım elde edilir.

2001 yılında; Kent Beck, James Grenning, Robert C. Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Arie van Bennekum, Andrew Hunt, Ken Schwaber, Alistair Cockburn, Ron Jeffries, Jeff Sutherland, Ward Cunningham, Jon Kern, Dave Thomas, Martin Fowler ve Brian Marick tarafından Çevik Yazılım Manifestosu oluşturulmuştur. Manifestoya göre; süreçler ve araçlar yerine bireyler ve etkileşimler, kapsamlı dokümantasyon yerine çalışan yazılım, sözleşme görüşmesi yerine müşteri işbirliği, planı takip etmek yerine değişime açıklık yazılımı geliştirmenin yoludur. Manifestonun bazı prensipleri vardır. Bunlar: En önemli öncelik müşteriyi memnun etmek ve erken sürekli yazılım teslimidir. Yazılım; yazılım sonlarına doğru bile olsa müşterinin avantajına yönelik değişim gerekliliklerini karşılamalıdır ve esnek olmalıdır. Birkaç hafta birkaç ay aralığında olmak üzere sık sık müşteriye çalışan yazılım teslim edilmelidir. Projede iş adamlarının ve proje ekibinin günlük olarak birlikte çalışması gereklidir. Projenin; motivasyon, güven ve destek veren bir ortamda çalışan, motive olmuş bireyler arasında yapılması gerekir. İletişimin en etkili yolu yüz yüze görüşmedir. Projenin birincil ölçütü çalışan yazılımdır. Çevik süreç sürdürülebilir geliştirmeyi teşvik eder. Sponsorlar, geliştiriciler ve kullanıcılardan süresiz olarak sabit bir hız ister. Sürekli teknik üstünlüğe dikkat ve güzel tasarım çevikliği geliştirir. Sadelik gereklidir. En iyi mimariler, gereklilikler ve tasarımlar kendi kendini organize eden takımlardan çıkar. Düzenli aralıklarla takım, nasıl daha etkili olabileceğini gözden geçirir ve verimliliği arttırmak için gerekli tutumu ve iyileştirmeleri yapar.

En yaygın uygulanan çevik metodolojiler şunlardır: Extreme Programming (XP), SCRUM, Agile Unified Process, Feature-Driven Development (FDD), Test-Driven Development (TDD), LEAN Development, Dynamic System Development Methodology (DSDM), Microsoft Solution Framework (MSF)’dür.

Extreme Programming (XP) Metodolojisi; 1999 yılında Kent Beck tarafından yapılmıştır. Extreme Programming takım çalışmasına ve müşteri memnuniyetine önem verir. Basit, etkili ve ekibin de yüksek üretkenlikte olmasını ister. Takım problemi etkili çözmek için kendi kendini organize eder. Extreme Programming yazılımı beş temel değer ile geliştirir. Bunlar; iletişim, basitlik, geri bildirim, saygı ve cesarettir. Projenin başarılı olabilmesi için Extreme Programming ekibin birbiri ile ve müşteri ile arasındaki yüz yüze ve eksiksiz iletişimi amaçlar. Sıkı iletişim olmaz ise önemli sorunlar ve anlaşmazlıklar çıkabilir. Basitlik için tasarım temiz ve sade tutulur, problemlere karmaşık çözümler aranmaz. O anki en basit çözüm kullanılmalıdır. Geri bildirim için proje ilk günden beri test edilir. Geri bildirim sayesinde oluşabilecek eksiklikler ve hatalar erkenden çözülür. Müşteri de proje grubunda sayılır ve 2-4 hafta aralıklarla program belirtilir. Proje grubu ve müşteri sıklıkla buluşup gelinen durumu incelerler. Saygı da her küçük başarıda ekipteki her birey ve müşteri arasında artar. Bu saygı temeli ile ekip yılmadan ve başarısızlıktan korkmadan, cesaretle proje geliştirmesine daha etkili rol oynar. Takım üyeleri korkmadan ve kısa zamanda müdahale etme cesaretini bulur.

Extreme Programming’in kolaylık sağlamak adına 12 pratiği bulunur. Bunlar; Planlama Oyunu, Ekipte Müşteri, Önce Test, Basit Tasarım, Çiftli Programlama, Sürekli Entegrasyon, Kısa Aralıklarla Sürümler, Yeniden Yapılandırma, Ortak Kod Sahiplenme, Metafor, Kodlama Standardı ve Haftada 40 Saat adlı pratiklerdir.

Planlama Oyunu; her yineleme için müşterinin de olduğu görüşmede işin ne kadar zamanda yapılacağı ile tahminde bulunmasıdır. Müşteri işi söyler ve yazılım ekibi işin yapılması için gereken süreyi belirtilen süre aralığına bağlı kalarak tahminde bulunur. Böylelikle iş süresi saptanır.

Ekipte Müşteride müşteri yazılım ekibiyle sadece toplantılarda değil yazılım gerçekleştirme sürecinde de aynı ortamda bulunur. Bu sayede yazılım ekibi geliştirme yaparken ihtiyacı olan bilgi olduğunda aynı ortamdaki müşteriden kolayca bilgiye erişir.

Önce Testte program yazılmadan test programı hazırlanır. Böylece ortaya çıkabilecek eksik ve hatalar daha program yazılmadan belirlenir. Güvenli yazılım amaçlanır.

Basit Tasarımda kısa sürede oluşan, yönetilen, değiştirilen ve anlaşılan, müşterinin de ihtiyaçlarını karşılayan en basit ve kolay yazılım gerçekleştirilir.

Çiftli Programlamada (Pair Programming) usta-çırak ilişkisi temel alınır. Yazılımcıların birbirlerine yardımcı olması, birleştirilmiş bilgi birikimi ve yeteneklerle projenin hızlı ve rahat geliştirilmesi amaçlanır. Farklı bakış açılarının aynı projede odaklanması ile hızlı sorun çözümü yapılır.

Sürekli Entegrasyonda (Continuous Intergration- CI) yazılım geliştirilirken yapılan yenilemeler be değişkenlikler o an sisteme entegre edilir ve günlük olarak test edilir. Böylece çıkabilecek sorunlar erkenden tespit edilir. Hemen entegrasyon edilmesi ile tüm yazılım ekibi sistemde yapılan değişiklikleri görebilir.

Kısa Aralıklı Sürümlerde proje iş parçalarına bölünür. Her parçanın kendine ait teslim tarihi (2-4 hafta) vardır. Teslim tarihinde müşteriye o parça teslim edilir. Müşteri her teslim yinelemesi ile çalışan, gelişen ve bütünleşen uygulamaya sahip olur.

Yeniden Yapılandırmada (Refactoring) ana fikir müşteri memnuniyetidir. Bunun için tasarım ve kod sürekli gözden geçirilir. Müşterinin istekler bu şartlar ile nasıl sağlanılabilir ve yazılım nasıl daha kolay ve basit yapılabilir diye bakılır.

Ortak Kod Sahiplenmede tüm yazılım kodu bireylere değil yazılım ekibine sahiptir. Böylece başka üyelerin yazdığı kod üzerinde iyileştirmeler ve geliştirmeler yapılabilir.

Metaforda (Benzetim) sistem parçalara ayrılır ve her parça bir sisteme benzetilir. Yap-boz gibi her parça biriminde yazılım yavaşça ortaya çıkar.

Kodlama Standardında yazılım ekibi belirlenmiş standart ve kurallara göre yazılımı geliştirir. Bu pratikte amaç kodun karmaşıklığını azaltarak anlaşılabilirliğini arttırmak ve kod bütünlüğü oluşturmaktır.

Haftada 40 Saat pratiğinde isminden de anlaşılabileceği üzere haftada 40, günde 8 saat çalışma süresi ile işler gerçekleştirilir. Fazla mesai sadece haftada bir gün vardır. Bu pratiğe göre 40 saatten fazla çalışma verimi arttırmaz, gereksizdir ve hata sayısında artış olabilir.

SCRUM Metodolojisi; 1990’ların ortalarında Jeff Sutjerland ve Ken Schawaher tarafından geliştirilmiştir. Bir proje yönetim yöntemidir ve sadece yazılım geliştirme süreci olarak düşünülmemelidir. Bu yöntem gözlemci, tekrara dayalı ve geliştirmecidir. Karmaşık, gereksinimlerin zor tanımlandığı projeler için kullanılır. Yenilemeler bir aydan fazla sürmez. Karmaşıklığı; şeffaflık, gözlem ve uyumlanma ile azaltmaya çalışır. Proje Sprint adı verilen 2-4 haftalık dönemler (birimlerde) gerçekleştirilir ve karmaşıklık parça parça çözülür.

İstenen ürünün hızlı, ucuz ve kaliteli üretilmesi için müşteri ile yüksek seviyeli gereksinim, içerik, ihtiyaç listesi oluşturulur. Buna Product Backlog denir ve merkezi belgedir. Sürekli güncellenen bir belgedir. Kullanıcı hikayelerinden oluşur. Kullanıcı hikayesi üç soruya cevap vermelidir. Kim? Neyi? Neden?

15-30 günlük proje dönemi içinde yapılacaklar listesine de Sprint Backlog denir. İş ve görev dağılımını bulundurur. Product Backlog ürün sahibini ilgilendirirken, Sprint Backlog SCRUM ekibini ilgilendirir.

Sprint Kalan Zaman Grafiği Sprint birimi içinde ne kadar iş yapıldığını ve yapılacağını gösterir. Ekibin zaman dağılımını iyi yapmasını amaçlar. Ekip her gün ne kadar iş yaptığı bilgisini girer. X ekseninde günlük zamanı, y ekseninde yapılacak kalan işleri gösterir.

SCRUM Metodolojisinin düzgün ilerleyebilmesi için belirli roller vardır. Bunlar; ürün sahibi (product owner), SCRUM yöneticisi (SCRUM master) ve SCRUM takımıdır (SCRUM team). Ürün sahibi; ürün geliştirme, maliyet, işlevsellik ve Product Backlog oluşturmaktan sorumludur. SCRUM yönteicisi; takım içerisindeki uyumu sağlar ve sorunları halletmekle görevlidir. SCRUM’ın başarılı olması için sürekli denetim yapar. SCRUM takımı iletişim halindeki ekiptir. Başlarında SCRUM yöneticisi bulunur.

SCRUM Metodolojisinde farklı toplantı tipleri de vardır.

Sprint (Koşu) Planlama toplantısında Product Backlog, risk değerlendirmeleri ve geliştirmeyle ilglili bilgiler ortaya çıkarılır. Ürünün maliyeti, pazarlanması ve dağıtımı hesaplanır. Takım ve yöneticisi belirlenir. Ne ve nasıl sorularına cevap aranır.

Günlük SCRUM Toplantısında SCRUM ekibi her gün, gün başlamadan maksimum 30 dakikalık toplantılar yapar. Bu toplantının üç temel konusu vardır. Dün ne yaptım? Bugün ne yapacağım? Beni engelleyen ne?

Sprint Gözden Geçirme Toplantısı her sprint’in sonunda veya başında yapılır ve sprint’in belirlenen hedef ve içeriklerde olup olmadığına bakılır veya belirlenir. Eğer eksiklik varsa ürün sahibi, Product Backlog’a tekrar kullanıcı hikayesi gönderir ve öncelik sırası verilir. Ekip de öncelik verilen gereksinimlerin ne kadarının yapılacağını belirler ve söyler.

SCRUM Metodolojisi karmaşık yeni projelerde veya başka strateji ile başlamış projeleri kurtarmak için kullanılabilir. Başarısı kanıtlanmış bir tekniktir ve önemli şirketler de kullanır. Bazıları; Microsoft, Google, IBM, Nokia, Siemens, Ikea, BBC’dir. Özetle dokümantasyonun sadece gerekli yerlerde olması, karmaşık sorunları parçalara ayırarak çözmesi sürekli yinelenen canlı bir ortam sunması, geri bildirimlere önem verip değişiklikleri sağlaması, iletişime önem vermesi, denetim halinde olması, çalışanların motivasyonu arttırması, müşterinin memnuniyetini sağlaması gibi özellikleri ile SCRUM günümüzde popüler bir strateji haline gelmiştir.

Yazılım yaşam döngü modelleri birbirinden tamamen ayrı değildir ve birlikte kullanılabilirler. Seçilecek modeller yazılımın türüne, alanına, maliyetine, ekip kapsamına ve verilen süreye göre değişebilir. Bazı modellerde dokümantasyon yokken, bazı modellerde en önemli faktördür, bazılarında ise sadece yeterli miktardadır. Bazı modeller karmaşıkken bazıları basitliği esas alır. Bazı modeller müşteriyi az ya da hiç dahil etmezken bazı modeller müşteriyi yazılım ekibinin bir parçası olarak görür. Bu özellikler ve günümüzün ihtiyaçları ile Çevik (Agile) Model en çok kullanılan model haline gelmiştir.

Yararlanılan kaynaklar:

<https://www.tutorialspoint.com/sdlc/sdlc_overview.htm>

<https://iskulubu.com/yazilim/yazilim-gelistirme-yasam-dongusu/>

<http://denizkilinc.com/yazilim-yasam-dongusu-temel-asamalari-software-development-life-cycle-core-processes/>

<http://w3.bilecik.edu.tr/bilgisayar/wp-content/uploads/sites/75/2019/02/ders-2.pdf>

<http://www.extremeprogramming.org/>

<https://moodle2019-20.ua.es/moodle/pluginfile.php/2213/mod_resource/content/2/agile-manifesto.pdf>

<http://denizkilinc.com/scrum-nedir/>

<https://tr.wikipedia.org/wiki/Scrum>

<https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>

Ezgi Hümeysa Savran- 210601700

LinkedIn : <https://www.linkedin.com/in/ezgi-h%C3%BCmeysa-savran-219574221/>

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

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