Yazılım yaşam döngüsü herhangi bir yazılımın ihtiyaç durumundan gerçekleştirilmesi ve sonrasındaki bakıma kadar olan sürece denir. Yazılım yaşam döngüsünün aşamaları şunlardır: Planlama, çözümleme. Tasarım, gerçekleştirim, bakım.

**Planlama:** İlk olarak gereksinim analizinin yapılmasıyla başlar. Gereksinim analizi müşterinin istekleri doğrultusunda ortaya çıkacak olan yazılımla ilgili ilk bilgilerdir. Bu adımda müşteri çok iyi dinlenmelidir. Müşteri ise isteklerini net ve eksiksiz olarak anlatmalıdır.

**Çözümleme:** Yazılımın işlevleri ve gereksinimlerinin ayrıntılı olarak çıkarıldığı aşamadır. Eğer varsa mevcut sistem incelenmesi yapılır. Bu aşamanın asıl amacı yazılım mühendisi gözüyle mevcut yapıdaki işlerin ortaya çıkarılması ve doğru algılandığından emin olmasıdır. Ayrıca bu aşamada temel olarak UML diyagramları çizilmeye başlanır. (Class, Use-Case gibi)

**Tasarım:** Bu aşama çözümlemeden sonraki aşamadır. Yazılımımızın veya sistemimizin tasarımları yapılır. Projeleri çizilir. Yazılımın modülleri bu aşamada karar verilir. Tasarım aşaması fiziksel tasarım ve mantıksal tasarım olmak üzere ikiye ayrılır. Fiziksel tasarımda yazılımı içeren bileşenler ve bu bileşenlerin ayrıntıları yer alır. Mantıksal tasarımda ise mevcut sistem yerine önerilen sistemin yapısı anlatılır.

**Gerçekleştirim:** Artık bu aşamada tasarımı bitmiş sistemin kodlanmaya başlandığı aşamadır. Kodlama işlemi planlama çözümleme aşamasında bulunan materyallerin yardımıyla yapılır. (Programlama dili, geliştirilme ortamı) Bu aşama üçe ayrılır: Kodlama, Test etme, Kurulum.

**Bakım:** Yazılım aşamasının ürün olarak sunulduktan sonra düzenli aralıklarla yazılıma yapılan hata giderici güncellemelerdir. Bu aşama yazılım tüm yaşamı boyunca devam eder. Bu yazılım yaşam döngüsünü daha aktif hale getirebilmek ve en iyi sonucu alabilmek için yazılım geliştirme modelleri kullanılabilir. Bu süreç modelleri temelde üçe ayrılır. Bunlar: Düzenleyici süreç modelleri, Birleşik süreç ve Çevik yazılım sürecidir.

***1-Düzenleyici Süreç Modelleri:***

**Kodla ve Düzelt:** Bu model genellikle resmi olmayan ürünün fikriyle başlar ve program ürün hazır olana kadar ya da gerekli zaman bitene kadar kodlanmaya devam eder. Avantajları arasında şunları gösterebiliriz: Planlamaya ihtiyaç yoktur, küçük çaplı projelerde ya da kısa ömürlük prototiplerde uygulanabilir, programlama aşamaları çabuk geçilir ve herkes bu modeli kullanabilir dolaysıyla uzman görüşüne ihtiyaç yoktur. Dezavantajları arasında ise kontrollü olmaması kaynak planlaması olmaması, bitiş süresinin belli olmaması gösterilebilir. Ayrıca hatayı bulmak zordur düzeltmesi maliyetlidir. Yazılan kodlar kullanıcının ihtiyacını karşılamayabilir.

**Çağlayan Modeli:** Şelale yöntemi olarak da bilinen bu model geleneksel yazılım metotlarında olduğu gibi lineer olarak işler. Her safhada bir önceki safhanın ürettiklerinin üstüne koyarak devam eder. Şelale modeli çok iyi hazırlanmış ve kısa sürede bitebilecek projelerde kullanılır. Avantajları: Kullanımı ve anlaması basit bir modeldir. Yönetimi basittir. Projenin aşamaları ayrı olduğundan dolayı iş bölümü ve planı projenin yönetimini çok kolay hale getirir. Dezavantajları ise şunlardır: Karmaşık ve nesne yönelimli projeler için uygun değildir. Uzun projeler için zayıf kalır. Müşteri memnuniyeti sağlamak çok zordur. Katı bir model olduğu için yapılacak bir değişim elverişsiz ayrıca çok maliyetlidir.

**V Modeli:** Bu modeli Şelale modelinin geliştirilmiş versiyonu olarak da düşünebiliriz. Doğrusal yönde ilerlemez. Onun yerine sürecin adımları kodlama evresinden sonra yukarıya doğru eğimler ve V şeklini oluşturur. Avantajları şunlardır: Projenin yönetimi tarafında takip basittir. Kullanımı kolaydır. Dezavantajları ise şunlardır: Aynı zamanda gerçekleştirilebilecek olaylara imkân zor tanınır. Risk çözümleme ile ilgili aktiviteler içermez. Yazılım zamanla evrimleşir. Bakım devam ettikçe ürünün gereksinimleri değişkenlik gösterir.

**Barok Modeli ve Gelişigüzel Model:** Bu iki modelde günümüzde artık kullanılmamaktadır. Barok modelinde döngü yoktur. Gelişigüzel modelde ise yöntem yoktur ve kişiye özel yapılır. Bu yüzden gözetilmesi zordur.

**Helezonik Model:** Bu model 4 aşamadan oluşur: Planlama, risk analizi, üretim ve kullanıcı değerlendirmesi. Planlamada her aşamada olan ara ürün için planlama yapılır. Risk analizinde ise riskler araştırılır, belirlenir, çözümü yapılır. Üretimde ara ürün üretilir. Kullanıcı değerlendirmesinde oluşturulan ara üründen sonra kullanıcıdan alınan geri dönütler değerlendirilir ve diğer aşamalara geçilir. Bu modelin dezavantajları şunlardır. Küçük ve az riskli projeler için çok maliyetli bir sistemdir. Kompleks içeriğe sahiptir ve dokümantasyonu çok fazladır.

**Artımsal Geliştirme Modeli:** Bu modelde kullanıcı istekleri aşamalı olarak karşılanacak şekilde birkaç yazılım ortaya konur ve gerçekleştirilir. Ara yazılımlar kullanıcı tarafından kullanılabilir. Yazılım açık mimaride herkesin anlayabileceği yapıda geliştirilir. Bu modelin avantajları şunlardır: Kullanıcı isteklerinin değişmesi, yazılımın tümünü etkilemeyebilir. Projeye başlamak için çok fazla maliyet gerekmez. Yazılımın birbiriyle ilgisiz olan parçaları bağımsız geliştirildiğinden daha hızlı yazılım geliştirilmiş olunur. Modelin zayıf tarafları şunlardır: Yazılım açık mimaride hazırlandığından bakımı oldukça zordur. Ek işlevleri yerine getiren yeni yazılım, bir öncekinin karşıladığı tüm işlevleri yerine getirmek zorundadır.

**Kodla ve Düzelt Yaşam Döngü Modeli:** Küçük programlar için kullanılabilir, direkt ürün gerçekleştirilir ve emeklilik safhası vardır. Büyük projeler için kullanılamaz, bakım az vardır ama zordur, ürünü hazırlar ve kullanıma sunarsınız.

Evrimsel Geliştirme Modeli: İlk tam ölçekli modeldir. Büyük firmalar için önerilir. Her aşamada üretilen ürün büyük bir işlevselliğe sahip olmaktadır. Model başarılıysa bunu ilk evrimin başarısına bağlayabiliriz.

***2-Çevik Modeller***

İteratif geliştirme temelli bir grup yazılım geliştirme metodolojisine dayanır. Burada gereksinimler ve çözümler farklı grupların ortak çalışmasıyla olgunlaşır. Çevik yazılım bir yandan bir değer sistemini, diğer yandan da somut yazılım metotlarını içerir. Bu modelin somut örnekleri arasında Extreme Programming, Scrum ve Lean Development bulunmaktadır.

**Extreme Programming (XP):** 4 temel maddeden oluşur: Basitlik, Cesaret, Geri Dönüş, İletişim. Basitlik, yazılan kodun yapılan işin sade anlaşılır ve karmaşık olmadan yapılmasını gerektirir. Fazlaca dokümantasyondan uzak durulur. Cesarette ise projelerin üstüne korkmadan gidilmelidir mottosu vardır. Bir kodun gerekirse tamamen silinip yeniden yazılması sağlanmalıdır. Geri Dönüş: Geri dönüşler ile oluşabilecek hatalar azaltılır/ortadan kaldırılır. Müşteri ile yazılım ekibi birbirleriyle iletişim halindedir. İletişim: İletişim, projelerde önemli sorunlardan birisidir. XP ise bunu aşmaya çalışmaktadır. Ekip içi iletişime önem verir ve artırılması için çalışır.

**Scrum:** Kompleks yazılım süreçlerinin yönetilmesi için kullanılır. Bunu yaparken bütünü parçalayan; tekrara dayalı bir yöntem izler. Müşteri ihtiyacına göre şekillendiği için müşterinin geri bildirimine göre yapılanmayı sağlar. İletişim ve takım çalışması çok önemlidir. 1990’ların başından beri kullanılmakta ve hatrı sayılır seviyede gün geçtikçe popülaritesi artmaktadır. Scrum sadece yazılım geliştirmeye değil, her yere uygulanabilir. Bu metodolojide bir yinelemenin tamamlanması 30 günden fazla sürmemekte ve günlük 15 dakikalık toplantılarla sürekli iş takibi yapılmaktadır.

Scrumda 3 temel kavram vardır. Roller, toplantılar ve bileşenler. Roller kategorisinde ürün sahibi, scrum yöneticisi ve scrum takımı vardır. Toplantılarda ise Sprint planlama, sprint gözden geçirme ve günlük scrum toplantısı vardır. Bileşenler/Araçlar kısmında ürün gereksinim dökümanı, sprint dökümanı ve sprint kalan zaman dökümanı vardır.

Ürün sahibi projenin iş değeri kısmından sorumludur yani projenin beynidir. Scrum yöneticisi ise takımı adapte eder. Scrum takımı ise tek bir hedefe ulaşmak için mücadele eden 5-6 kişilik takımdır.

Scrumun olmazsa olmazı toplantılardır. Her gün toplantılar yapılır geçmiş gün ve bugünle ilgili yapılanlar ve yapılacaklar konuşulur. Her sprint için de bir gözden geçirme toplantısı yapılır.

Bileşenler ve Araçlarda ürün gereksinim dokümanları oluşturulur. Bu dokümanda kısaca yapılması gereken şeyler yazılıdır. Print dokümanı ürün gereksinim dokümanına takiben oluşturulur ve amacı her sprintin ona uygun ayarlanmasıdır. Sprint Kalan Zaman Grafiği ise yapılan işin ne seviyede olduğu ve aslında planlanan zaman göre nerede olduğunu belirlemek için hazırlanır. Sprint backlog sadece takım tarafından değiştirilir.

***SCRUM NEDEN GÜNÜMÜZDE BU KADAR POPÜLER?***

Çünkü zamandan ve paradan büyük ölçülerde tasarruf edilir. Son gelişmelere ve yüksek teknolojilere rahatlıkla uyum sağlar. Kompleks gözüken ve gereksinimleri tam belli olmayan projeler için çok idealdir. Ekip içinde iletişim her zaman üst seviyededir. Bu sayede hatalar hızlıca fark edilip düzeltilir. Kullanıcıdan gelen geri bildirimler ile sorunların büyük oranla azalır. Diğer yazılım geliştirme modelleri gibi yinelemelidir. Sürekli değişen gereksinimlere hızlı reaksiyon verir.

***HANGİ YAZILIM GELİŞTİRME MODELİ DAHA KULLANIŞLI?***

Barok ve gelişigüzel modelleme günümüzde kullanılmaz çünkü yinelemeli değildir ve dokümantasyon içinde barındırmaz.

Şelale modeli kullanılması basit ve iş planlaması detaylıdır. Küçük ve iyi belirlenmiş projelerde iyi iş yapar. Eskiden yaygındı fakat günümüzde popülerliği azaldı çünkü iletişim zayıflamaya başlamıştır ve büyük projeler için zayıf kalmaya başlamıştır. Değişimlere kapalıdır.

Evrimsel Geliştirme Metodunda kullanıcılar gereksinimleri daha iyi anlarlar, hatalar azalır ancak sürecin görünürlüğü azdır ve bakımı zordur.

Belirsizliklerin az olduğu, iş tanımlarının belirgin olduğu bilgi teknolojileri projeleri için V Modeli uygun bir modeldir.

Eğer büyük kitlelere ulaşmak istiyorsak bu projeler için evrimsel geliştirme modeli kullanılabilir.

Eğer projemiz büyük maliyetli ve uzun sürsün istersek artımsal geliştirme veya spiral model uygundur.

Orta ve küçük çaplı projelerde çevik yazılım modelleri çok uygundur.

Eğer projemiz kişiye özel olacaksa ve zaman sorunumuz yoksa kodla ve düzelt modelini kullanabiliriz.

Küçük ve özellikleri yeterince iyi belirtilmemiş projelerde Çağlayan (Şelale) modeli uygundur.

Kullandığım Kaynaklar:

* <https://caglartelef.com/yazilim-yasam-dongusu/>
* <https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>
* <https://en.wikipedia.org/wiki/Scrum_(software_development)>
* <https://www.quora.com/Why-is-the-Scrum-process-so-popular-in-the-software-industry>
* Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 2. ve 3. Hafta Sunumları

İhsan Umut Akpolat

180601050