YAZILIM YAŞAM DÖNGÜ MODELLERİ

Bir yazılımın ortaya çıkarılması, geliştirilmesi ve bakımı için kullanılan doğrusal olmayıp döngü içinde tekrar eden adımlar bütününe yazılım yaşam döngüsü denir. Bu döngü sırasında geriye dönmek ve yinelemeli olarak ilerlemek söz konusu olabilir. Yazılım yaşam döngüsünün beş temel aşaması vardır bunlar; planlama, çözümleme, tasarım, gerçekleştirim ve bakım aşamalarıdır.

1. Planlama Aşaması: Projenin bel kemiği sayılabilecek bir aşamadır. Kullanıcının gereksinimlerinin anlaşılması beklentisini karşılama açısından önemlidir. İş bölümü ve görevlendirmeler yapılır. Projenin ya da iş fikrinin uygulanabilirliğini ortaya koyan fizibilite çalışması ve planlama bu bölümde yapılır.
2. Çözümleme aşaması: Projenin hangi amaçla yapıldığı ve projenin gereksinimleri nelerdir, risk durumunun belirlenmesi gibi maddeler belirlenir, ayrıntılı inceleme yapılır. Bu aşamaya analiz aşaması da denebilir.
3. Tasarım aşaması: Yazılım projesinin nasıl ilerleyeceği tasarlanan aşamadır. Projeler ve temel yapı oluşturulur. Sistem tasarımı ve ayrıntılı yazılım tasarımı adı verilen iki aşaması vardır.
4. Gerçekleştirim: Yazılım kodlanır, birleştirilir, hata olma durumuna karşın test edilir ve kurulumu yapılır.
5. Bakım aşaması: Yazılımın hataları giderilir ve yeni modüller eklenir bu aşama sonlanmaz bir döngü içinde devam eder.

Yazılım yaşam döngü modelinde pek çok model ele alınabilir. Bu modeller yazılım projesinin büyüklüğü, yazılım projesinin hangi amaçla kim tarafından kullanacağı gibi durumlar nedeniyle birden fazladır. Bunlardan: Gelişi Güzel Model, Barok Modeli, Çağlayan (Waterfall) Modeli, V Modeli, Sprial Model (Helezonik Model); mevcut geleneksel modellere alternatif olarak oluşturulan çevik yazılım sürecinden: Uçdeğer Programlama (Extreme Programming-XP) ve Scrum örnek verilebilir.

1.Gelişi Güzel Model: 1960'larda kullanılan bu modelde belirli bir yöntem kullanılmaz bu nedenle bu yönteme model demek doğru sayılmaz. Sadece geliştiren kişiye bağlı, kişisel şekilde yapılır. Bu şekilde geliştirilen yazılımların izlenebilirliği kolay bir şekilde sağlanamaz ve bakımı zordur.

2.Barok Modeli: Bu model 70'li yılların ortalarına doğru kullanılmaya başlamıştır. Yaşam döngüsü çekirdek süreçleri doğrusal bir şekilde gerçekleştirilir. Belgeleme işlemi bu modelde güncel modellerinden farklı olarak ayrı bir süreçtir. Yazılımın geliştirilmesi ve test edilmesinden hemen sonra belgeleme yapılır. Bu modelde olan zayıf bir taraf da adımlar arası geri dönüş tanımlı değildir.

3.Çağlayan Modeli: Bu modelde yazılım yaşam döngüsü adımları en baştan en sona bir kez uygulanır. Tanımlanması çok iyi olan ve üretiminde fazla zaman gerektirmeyen projelerde uygulanır. Günümüzde bu modelin kullanımı gittikçe azalmaktadır. Çağlayan modelinde barok modelindeki gibi belgelendirme kısmı ayrı bir adım değildir üretimin içinde olan bir parçasıdır. Bir sonraki adım önceki adım tamamen tamamlandıktan sonra başlar. Her aşamanın sonucu belgelenir. Oluşturulan belgeler kendinden sonraki adımda girdi olarak kullanılır. Eğer herhangi bir safhada belgelendirme olmayıp test edilmemişse o safha tamamlanmamış kabul edilir. Gerektiği geliştirme aşamasında tekrarlamalar mümkündür. Çağlayan Modelinde dikkat

edilmesi gereken hususlar; en ince detayların bile tasarımda yansıtılabilmesi için müşteri gereksinimlerinin ayrıntılı bir şekilde ortaya koyulması ve özellikle uzun zamanda bitirilecek projelerin gereksinimlerinin değişebilecek olması ayrıntısını atlamamaktır. Yeniliklere açık olamayışı sebebiyle sabit projelerde kullanılması daha uygundur.

3.1.Çağlayan Modeli Avantajları:

- Kullanımı ve yönetimi oldukça kolaydır. Adımları müşteriler ve son kullanıcılar tarafından kolay kavranabilirdir.
- Daha uzak adımlarda olması nadir olmakla birlikte tekrarlamalar, bir sonraki ve bir önceki adımlarla gerçekleşir.
- Gereksinim modelinden sonra sağlam bir temel ortaya çıkar.
- Proje yöneticilerine iş dağılımı konusunda kolaylık sağlar.

3.2.Çağlayan Modeli Dezavantajları:

- Uzun zamanda son kullanıcıya ulaşır, karmaşık projeler için uygun olmamakla birlikte uzun projeler için zayıf bir yapısı vardır.
- Projelerdeki değişim ve gelişimlere karşı uygun bir model değildir.
- Kullanıcı sürecin içinde yoktur. Gereksinimler iyi tanımlanmamışsa veya müşterinin ne istediğinin anlaşılmadığı bir projede proje iptali gibi zor durumlar ortaya çıkabilir bunun nedeni model safhalarından oluştuğundan son safhada tamamlanmasıdır.
- İki ya da ikiden daha fazla önceki aşamalara gitmek oldukça maliyetlidir.

4. V Modeli: Bu modele çağlayan modelinin gelişmiş hali denebilir. Çağlayan Modelinden farklı olarak üretim ve sınama işlemlerinin ne zaman yapılacağı öne çıkarılır. V'nin sol tarafı projenin üretim kısmıyla sağ tarafı ise sınama işlevleriyle ilgilidir. Eğer sınama işleminde hata bulunursa bu modelde nereye dönülmesi gerektiği de belirtilmektedir. Örnek verilecek olunursa sağ tarafta bir sınama işleminde bir hata varsa sol tarafta o hizadaki adıma dönülür. V modelinin üç temel çıktısı: Kullanıcı Modeli, Mimari Model ve Gerçekleştirim Modelidir. Genellikle büyük projelerde kullanılmakla beraber bu modelin uygun olduğu projelerin belirsizliklerinin az olması gerekir. Bilgi Teknoloji projeleri bu model için uygundur.

4.1. V Modeli Avantajları:

- Kullanıcı bu modelde projeye katkı sağlar.
- Projenin yönetimi ve takibi kolaydır.
- Tüm teslim edilebilir ürünlerde verification ve validation planları erkenden uygulanır.
- Kolay kullanılır.

4.2. V Modeli Dezavantajları:

- Adımlar arasında tekrarlama kullanılmaz.
- Risk çözümleme açısından zayıftır.
- İş ve ürün gereksinimlerin değişkenlik gösterme ihtimali vardır.

5.Sprial Model: Risk analizinin oldukça önemli olduğu bir modeldir bu durum potansiyel zorlukları en aza indirir. Süreç sıralı aktiviteler şeklinde değil sprial şekilde gösterilir. Bu sprial üzerindeki bir halka bir fazı temsil eder. Dört temel aşamadan oluşur bunlar: her aşamanın planladığı planlama aşaması, risklerin belirlenip çözüldüğü risk analizi aşaması, üretim aşaması ve ara ürünün kullanıcı tarafından yapılan değerlendirme, sınama gibi işlemleri barındıran kullanıcı değerlendirme aşamasıdır.

5.1.Sprial Model Avantajları:

- Kullanıcıların sistemi erkenden görmesi sağlanır.
- Hataların olabildiğince erken giderilmesine dikkat edilir.
- Geliştirme kısmı küçük parçalara bölünüp risk durumu yüksek olan parçalar önce gerçekleştirilir.

5.2.Sprial Model Dezavantajları:

- Az riskli projeler için pahalıdır.
- Karmaşık olmakla beraber sprial sonsuza girme durumu vardır.
- Çok fazla belgeleme gerektirir bunun nedeni çok fazla ara adımın olmasıdır.

Çevik Modeller: 1990'larda kullanılmaya başlayan bu model türleri üretim alanında verimliliğin arttırılması ve yazılımlara daha esnek ve kullanışlı bir biçimde uygulanabilme oranını arttır. 2001 yılında yayınlanan çevik yazılım geliştirme manifestosuna göre bu modellerde bireyler arasındaki etkileşimi, çalışan yazılımı, müşteriyle iş birliğini ve plana bağlı kalmaktansa değişimi önemsemeyi önemli kılmıştır. Bu model çoğunlukla karmaşık ve uzun projelerde kullanılır.

1.Extreme Programming (XP): 1996 yılında Kent Beck ve arkadaşları tarafından kurulan bu yazılım geliştirme disiplininin; basitlik, cesaret, geri dönüş ve iletişim olmak üzere dört temel maddesi vardır.

Basitlik: Çok uzun belgelemelerden uzak durularak anlaşılır ve düzenli bir şekilde kod oluşturulmasıdır. Karmaşık çözümler XP modeli için uygun değildir.

Cesaret: Başarısızlıktan çekinmekten ziyade başarısızlıklara sebep olabilecek nedenlerin üzerine gitmek ve başarısızlığı olabildiğince hızlı düzeltmek temel prensiptir. Cesur bir şekilde hareket edilmeli gerektiğinde çekinmeden kodun tamamının silinip yenisinin yazılabilmesidir.

Geri dönüş: belirli aralıklarla geriye dönüş yapılarak oluşan hatalara bakılır böylelikle gelecekte bu sorunların tekrarlanması önlenir.

İletişim: XP de ekibin sağlıklı ve güçlü bir şekilde iletişimde olmalıdırlar. İletişim mutlaka yüz yüze olmalıdır böylelikle projelerdeki en önemli sorunlardan biri olan iletişim sorunu en aza indirilir.

XP'nin bütün olarak bakılması gereken on iki temel pratiği vardır:

Planlama Oyunu: Müşterilerle beraber çalışılarak gereksinimlerin belirlenip planlanması sağlanır.

Ekipte Müşteri: Müşterinin erken geribildirim yapma amacıyla ekiple beraber çalışmasıdır.

Önce Test: Testin öncelikli olmasıdır yazılım tüm testleri geçtiğinde duyulacak güven oldukça önemlidir ve sürekli entegrasyonun gerçekleşmesine de yardımcı olur.

Çiftli Programlama: İki kişi olarak çalışılmasıdır. Bir kişi metottan diğer kişi stratejik düşünmeden sorumludur. Roller belli aralıklarla değişir. Herkesin tüm bileşenlerden haberdar olmasını sağlar.

Basit Tasarım: Programlama çift kişi yapılacağı için buna göre anlaşılır bir tasarımı olmasıdır.

Sürekli Entegrasyon: Bir testin ilerde üç dört teste bağlı olmasıdır bu uyumluluk sorunlarını erkenden önlemeyi sağlar.

Kısa Aralıklı Sürümler: 2 ya da 6 aylık sürümlerin çıkması.

Yeniden Yapılandırma: Testler yazıldığında tasarımın karmaşıklaşmasına karşın tasarımda iyileştirmenin yapılması.

Ortak Kod Sahiplenme: Tüm takımın her bilgiye hâkim olmasıdır böylelikle kaynak devir teslim işlemi kolaylaştırılır.

Metafor: Sistemde alt sınıfları gösteren basitleştirilmiş sistem mimarisidir.
Kodlama Standardı: Kodda ortak anlayış kurmak amacıyla standartlar oluşturulmasıdır
Haftada 40 Saat: ortalama 8 saat 5 gün çalışılmasıdır.
2.Scrum: Popüler olan bir yazılım geliştirme modelidir.1990'ların ortalarına doğru Jeff Sutjerland ve Ken Schawaber tarafından geliştirilmiştir. Proje yönetimi ve planlamaya önem verir. Mühendislik detayları bu modelde bulunmaz böylelikle her projeye uygulanması mümkündür. Kompleks bir yazılımı birimlere böler ve yinelemeli olarak geliştirir. Bu bölünen parçaların her birine "sprint" adı verilir. Proje ilerlemesi, gelişimi ve problemlerinin herkese açık olmasını gerektiren; şeffaflık, proje ilerlemesinin düzenli bir şekilde kontrol edilmesini sağlayan; gözlemleme ve uyarlanacak değişikliklerle uyumlu olması açısından; uyarlama adı verilen üç temel prensip üzerine kurulmuştur. Karmaşık projeler için en uygun metodoloji budur çünkü bu metodolojide gereksinimler kolaylıkla tamamlanır. Scrum modelinde roller, toplantılar ve bileşenler olmak üzere üç temel kavram vardır.
1.Roller:
Ürün sahibi: Müşterinin kendisi veya müşteriyi temsil eden müşterinin gereksinimlerini aktaran kişidir. Ne istendiğini ve ürünün amaçlarını yazılım ekibine aktarmakla sorumludur.
Scrum Yöneticisi: Kurallar, teoriler ve pratikler konusunda bilgili ve bunların sağlanmasında görevlidir. Takımlardaki sorunları ön gören bunları çözmeye çalışan, çalışanlara yardımcı olan organizasyonla görevli olan kişidir. Takım içi düzeni sağlar. Takımın yöneticisi değil lideridir. Scrum Modelinde klasik anlamda yönetici deneyimsiz bir ekipte sorun yaratabileceğinden veya takımın özerkliğini bozabileceğinden dolayı yoktur.
Takım: Beş dokuz arasında kişiden oluşan yazılım geliştirme ekibidir. Herkes her işi yapabilir durumdadır ve zamanla rol değişimi olabilir. Kendi kendilerini yönetirler iş dağılımını beraber yaparlar.
2.Toplantılar:
Sprint Planlama: Hedef tanımlanır. Gereksinimlerin listesi oluşturulup bazıları minik görevlere ayrılır. Takımlar belirlenir. Takımdaki kişiler kendi aralarında görev paylaşımı yapar. Maliyetler hesaplanır. Risk değerlendirmesi yapılır.
Sprint Gözden Geçirme: Sprintlerin sonunda yapılır. Ürün gözden geçirilir. Ürün değerlendirmesi yapılır. Bir hata varsa düzeltilir.
Günlük Scrum Toplantısı: Genelde sabahları ayak üstü on beş dakika yapılan toplantılardır. Günü planlamak amacıyla yapılır. Takımdakiler varsa karşılaştıkları problemleri açıklarlar.
3.Bileşenler:
Ürün Gereksinim Dokümanı: Yapılacaklar listesi gibi düşünülebilir. Gereksinimlere göre bu dokümana ekleme çıkarma işlemleri yapılabilir. Kullanıcı hikayelerinden oluşur.
Sprint Dokümanı: Ürün gereksinim raporundaki iş ve görevler öncelik sırasına göre sıralanır. İşlerin zaman çizelgesi oluşturulur.
Sprint Kalan Zaman Grafiği: Sprintin günlerini ve kalan işi gösteren grafiktir. Şeffaflık sağlar.
Scrum uzmanlık gerektiren ve maliyeti yüksek bir model olmasına rağmen modeli temel prensipleri, hızlı sorun tespitleri, ekipler içinde esnekliklerin olması, sürekli kendini geliştirmeyi ön planda tutması, işlerin verimliliğini arttırması, basit kurallardan oluşması gibi nedeler sayesinde günümüzde oldukça popülerdir. Microsoft, IBM, Google ve Yahoo gibi büyük firmalarda kullanıldığı gibi küçük firmalarda kullanılabilen bir metodolojidir.

Günümüzde Barok ve Gelişi Güzel Model yinelemeli olmama veya belgelendirme yönünden zayıf olması gibi özelliklere sahip olduğundan yaygın olarak kullanılmasa da diğer proje yönetim şekillerinin kendine göre avantajları ve dezavantajları vardır. Uygulama açısından Sprial Modeli karmaşıkken, Çağlayan Modeli, V modeli, Çevik modeller kolay yapıdadır; başarı garantisi Çağlayan Modelde düşükken V Modelinde orta düzeyde, Sprial Model ve Çevik Modellerde çok yüksek seviyededir. Risk duyarlılığı Çağlayan Modelde yüksek; Sprial Model, V modelinde düşük Çevik Modellerde ise azaltılmıştır. Zamanlama açısından karşılaştırılırsa Çağlayan Modeli için çok uzun bir zaman gerekirken V Modelde ve Sprial modelde, Çağlayan kadar olmasa da yine de uzun bir zaman gerekir; Çevik Modellerde daha kısa bir süre yeterlidir.

Sonuç olarak yazılım yaşam döngüsü modelleri izlenebilir ve tekrarlanabilir projeler için gereklidir ve projenin gereksinimleri, projelerin büyüklüğü, karmaşıklığı, ekip bilgileri gibi bilgiler dikkate alınarak seçilmelidir. Her biri farklı alanlarda avantajlar ve dezavantajlar içerir.

SUEDA AHSEN GÖKALP

YARARLANILAN KAYNAKLAR:

- https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/
- https://ybsansiklopedi.com/wp-content/uploads/2015/08/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Modelleri-Yaz%C4%B1l%C4%B1m-Ya%C5%9Fam-D%C3%B6ng%C3%BCs%C3%BCSDLCYBS.pdf
- https://caglartelef.com/yazilim-yasam-dongusu/
- https://talentgrid.io/tr/yazilim-gelistirme-modelleri/
- ZEKERİA ANIL GÜVEN YAZILIM MÜHENDİSLİĞİ TEMELLERİ DERS NOTLARI
- https://furkanalniak.com/yazilim-muhendisligi-yazilim-surec-modelleri/
- https://blog.emrahkahraman.com.tr/xp-extreme-programming-pratikleri/
- https://www.bilgisayarmuhendisleri.com/sayfa.aspx?s=52
- https://agilemanifesto.org/iso/tr/manifesto.html
- https://www.mshowto.org/agile-ve-scrum-nedir.html
- https://medium.com/@ahmetuyar/extreme-programming-xp-nedir-ddc003a515c4
- https://www.savassakar.com/pmp-hazirlik-cevik-yaklasimlar-ekstrem-programlamaxp/
- https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742
- https://ofcskn.com/tr/yazilimda-sdlc-modelleri-nelerdir
- https://www.abprojeyonetimi.com/scrum-proje-yonetimi-nedir/
- https://binyaprak.com/yazilar/scrum-serisi-2-scrum-terimler-ve-rolleri
- https://netuce.com/scrum-nedir-scrum-rollerini-taniyalim/