Yazılım Yaşam Döngüsü ve Modelleri

Selim Eren Kaya
220601052
İzmir Bakırçay Üniversitesi, Bilgisayar Mühendisliği

Özet:

Bu yazı kapsamında, yazılım yaşam döngüsü olarak bilinen kavramı ve bu kavrama bağlı olaraktan literatürde sıkça kullanılan bazı yazılım yaşam döngüsü modelleri incelenmiştir. Bu modeller gelişigüzel model, barok modeli, şelale modeli (waterfall model), V süreç modeli, helezonik model (spiral model), artımsal geliştirme modeli, kodla ve düzelt modeli, evrimsel geliştirme modeli, extreme programming (XP) ve scrumdur. Modellerin genel özellikleriyle birlikte hem iyi hem de kötü yanlarına değinilmiştir.

1. Yazılım Yaşam Döngüsü Nedir?

Yazılım yaşam döngüsü, planlamadan teslimata kadar bir yazılım projesinin yaşamış olduğu tüm aşamalara ve bu aşamalardan oluşan döngüye denir. Bu aşamaların sayısı değişebileceği gibi genel anlamda aşağıdaki gibi sıralanabilir.

1-Planlama
2-Analiz
3-Tasarım
4-Geliştirme
5-Entegrasyon ve testler
6-Uygulama
7-Bakım

1.1. Planlama

Yazılım yaşam döngüsünün başladığı yerdir. Bu süre zarfında projenin fizibilite çalışması yapılır, temel gereksinimler belirlenir ve proje planlaması yapılır. Aşamalar arasındaki en önemli aşamadır.

1.2. Analiz

Yazılım yaşam döngüsünün en önemli aşamalarından biri olan analiz sürecinde tüm proje fonksiyonlarına kesin olarak karar verilir. Alınan bu kararlar, sistem ihtiyaçlarının netleştirilmesine ve ilgili taleplerin hazırlanmasına olanak sağlar. Diğer bir deyişle, analiz yöntemi projenin her yönünü ortaya koymaktadır.

1.3. Tasarım

Analiz çalışmasının sonuçlarına göre proje bileşenlerine ayrılır, adım adım proje eylemleri belirlenir ve bir proje planı oluşturulur. Proje planına ek olarak tasarım belgesi hazırlanmalıdır. Proje, sistem tasarımı, veri modeli ve kullanıcı arabirimi tasarımları tasarım belgesinde yer alır. Tasarım belgesinin amacı, yazılımcının yazılımı oluştururken başvurabileceği teknik dokümantasyonu sağlamak ve daha sonra üzerinde çalışacak yeni yazılımcılar için projeyi daha kolay anlaşılır kılmaktır.

1.4. Geliştirme

Planlama, analiz ve tasarımın tamamlandığı ve yapılacak belirli eylemlere karar verildiği projenin geliştirme aşamasındadır. Şu anda daha fazla analiz yapılmamalıdır; bunun yerine proje, tasarım aşamasında oluşturulan planlamanın kısıtlamaları dahilinde

ilerlemelidir. Yazılım projeleri için kodlama kısmı olarak düşünülebilir. Bir bakıma daha önceden senaryosu yazılmış filmin çekilmeye başlandığı aşamadır.

1.5. Entegrasyon ve testler

Entegrasyon ve testler, yazılım oluşturma işlemi tamamlandıktan sonra müşteriye verilmeden önce gerçekleşir. Bu aşama tamamlandıktan ve hatalar düzeltildikten sonra proje müşteriye teslim edilir.

1.6. Uygulama

Proje artık bir canlı haline gelir ve kullanılmaya başlanır. Kullanıldıkça sorunlar ortaya çıkar. Bu sorunların karşılığında yeni fikirler, yeni ihtiyaçlar ortaya çıkar ve bunlar dikkate alınarak bir inceleme başlar. İncelemenin ardından bakım aşamasına geçilir.

1.7. Bakım

Proje yayınlandıktan sonra ortaya çıkan hataları düzeltmeyi, yazılımı geliştirmeyi ve yeni özellikler eklemeyi içerir. Bu talepler, bu aşama sırasında kullanıcılardan gelen veriler doğrultusunda yerine getirilir.

2. Yazılım Yaşam Döngüsü Modelleri Nelerdir?

Günümüzde yazılım yaşam döngüsünde çeşitli modeller dikkate alınır. Çeşitli modellerin olmasının sebepleri yazılım projesinin boyutu, kimler için kullanılacağı gibi sebeplerle açıklanabilir. Bu modellerden bazıları aşağıdaki listede görülebilmektedir.

1-Gelişigüzel Model
2-Barok Modeli
3-Şelale Modeli (Waterfall Model)
4-V Süreç Modeli
5-Helezonik Model (Spiral Model)
6-Artımsal Geliştirme Modeli
7-Kodla ve Düzelt Modeli
8-Evrimsel Geliştirme Modeli
9-Extreme Programming
10-SCRUM

3. Gelişigüzel Model

Bir yöntem veya plan yoktur. Kişiye özel yapılır ve gözetilmesi oldukça zordur. 1960'larda kullanımdaydı.

4. Barok Modeli

Yaşam döngüsünün aşamaları doğrusal bir şekilde gözden geçirilir. Döngü yoktur ve belgeleme süreçleri ayrı ayrı ele alınır. 1970'lerde kullanımdaydı.

5. Şelale Modeli (Waterfall Model)

Mevcut güncel yaşam döngüsü modellerinin bu modelin temelleri üzerine inşa edildiğini söyleyebiliriz. Bir zamanlar en yaygın yaşam döngüsü modeliydi. Bu modeldeki her aşama en az bir kez gerçekleştirilir. Şelale modeli, net bir kapsamı olan ve hızlı bir şekilde tamamlanabilecek projelere uygundur. Şelale modeli, barok ile karşılaştırıldığında belgeleme sürecin içerisindedir. Bir sonraki aşamaya geçmeden önce bir öncekinin bitirilmesi gerekir. Bu model kullanılırken karşılaşılabilecek sorunlar aşağıdaki gibidir. Yazılımın geliştirilmesi ve piyasaya sürülmesi genellikle uzun bir süreçtir, ancak şelale yaklaşımı buna uygun değildir.

Kullanıcı geliştirme sürecine dahil olmadığı için, potansiyel bir sorun çok geç olana kadar belirlenemeyebilir, bu da yazılımın maliyetini artırır ve süreci uzatır.

6. V Süreç Modeli

V-modeli geliştirilmiş bir şelale modeli olarak düşünülebilir. Az sayıda belirsizlik ve net iş tanımları olduğunda, bu model kullanılır. Planlama, gereksinimleri belirleme ve hem üst hem de alt düzey tasarımları oluşturma vardır. Daha yüksek bir düzeyde, tasarım daha genel olarak görme ile oluşturulur. Daha karmaşık olan daha düşük seviyeli tasarımların oluşturulması, yüksek seviyeli tasarımları takip eder. Alt katmanlar açılır ve girdileri, çıktıları, beklentileri yazılır. Bundan sonra, kodlama aşaması başlar. Proje yönetimi, aşağı doğru akan bir süreç yönetimi vardır. Bu noktadan itibaren yükselmeye başlar. Birim testleri yapılır. Üretilen her modül test edilir. Daha sonra bu modüller ile düşük seviye tasarımlı ürünlerin birlikte çalışabilirliğini sağlamak için entegrasyon testleri yapılır ve kabul testleri aşamasında müşteri ile görüşülerek müşterinin onaylayıp onaylamadığı belirlenir. Başvuru müşteri tarafından incelenir. Ardından bakım başlar. Bu modelde karşılaşılan sorun ise şudur: Aşamalarda tekrar bulunmaz ve risk çözümleme için ayrılan bir yer yoktur.

7. Helezonik Model (Spiral Model)

Helezonik modeli, risk analizi ve prototip geliştirmeye öncelik verilmesi bakımından diğerlerinden farklıdır. Risk analizi merkezdedir, bu nedenle hataları erken yakalama şansı sunabilir. Prototipleme her aşamada yapılır ve kullanıcıya her aşamada yazılım ürününün bir parçasını görme şansı verir. Bu sayede sorunların çözülmesini daha kolay hale getirir. Helezonik modelin sorunları şunlardır: Küçük ve düşük riskli projeler için çok pahalı bir sistem olması, karmaşık bir içeriğe sahip olması, uzun sürmesi ve fazla dokümantasyondan oluşması olarak sıralanabilir. Temel olarak 4 aşaması vardır: Planlama, Risk Analizi, Üretim, Kullanıcı Değerlendirmesi.

7.1. Planlama

Her aşamada olan ara ürün için bir planlama yapılır.

7.2. Risk Analizi

Risklerin araştırılması, belirlenmesi ve çözülmesi yapılır.

7.3. Üretim

Ara ürünün/ürünün üretilmesidir.

7.4. Kullanıcı Değerlendirmesi

Oluşturulan ara ürünün sonucunda kullanıcıdan alınan geri dönüşlerin değerlendirilerek diğer aşamaya geçilmesidir.

8. Artımsal Geliştirme Modeli

Artımsal geliştirme modelinde proje parçalara ayrılır ve bu bileşenler kullanıcı tarafından önceliklendirilir. Bu işler tamamlandıktan sonra birer birer ara ürün oluşturulur ve kullanıcı bu ara ürünleri kullanır. Her seferinde ara ürünler bir öncekinin üstüne bir şeyler katarak çıkartılır. Yani bu modelde üretim kısmı devam ederken bir yandan da kullanım kısmı devam eder. Artımsal geliştirme modeli, tamamlanması biraz zaman alabilen ve tamamlanmış sonucun sınırlı işlevsellikle çalışabileceği yazılımlar için uygundur. Bu yaklaşımla bir sistemin arızalanma olasılığı azaltılır ve ara ürünler yazılım geliştirme sürecinde önemli bir rol oynar. Sorunları ise şunlardır: Her parçanın kendini tekrar etmesi yasak olduğundan, bir ara ürün bitip diğeri başlayana kadar herhangi bir değişiklik

yapılamaz ve parçalar oluşturulmadan önce bu sistemin tam olarak detaylı bir şekilde tanımlanması gerekir.

9. Kodla ve Düzelt Modeli

Küçük projeler için kullanılabilir, direkt olarak ürün gerçekleştirilir ve emeklilik safhası vardır. Büyük projeler için kullanılamaz, bakım az vardır ama zorlayıcıdır, ürünü hazırlar ve kullanıma sunarsınız.

10. Evrimsel Geliştirme Modeli

İlk tam ölçekli modeldir. Büyük alanlara yayılmış, büyük firmalar için önerilir. Her aşamada üretilen ürün tam işlevselliğe sahiptir. Modelin başarısı ilk evrimin başarısına bağlıdır.

11. Extreme Programming (XP)

Extreme Programming (XP), geliştirme ekibi için daha yüksek kaliteli yazılım ve daha yüksek yaşam kalitesi üretmeyi amaçlayan çevik bir yazılım geliştirme çerçevesidir. XP, yazılım geliştirmeye uygun mühendislik uygulamalarına ilişkin çevik çerçevelerin en spesifik olanıdır. Kent Beck ve arkadaşları tarafından 1996 yılında kurulmuştur. 4 temel maddeden oluşur: Basitlik, Cesaret, Geri Dönüş, İletişim.

11.1. Basitlik

Yazılan kodun ve yapılan işin sade, anlaşılır ve karmaşık olmadan yapılmasını gerektirir. Uzun uzun dokümantasyondan uzak durulur.

11.2. Cesaret

Yapılan işte cesur davranılmalı ve projeler korkusuzca sürdürülmelidir. Gerekirse bir kod tamamen silinmeli ve yeniden yazılmalıdır.

11.3. Geri Dönüş

Geri dönüş, hataları en aza indirmeye ve hatta tamamen ortadan kaldırmaya yardımcı olabilir. Yazılım personeli ile müşteri arasında iletişim vardır.

11.4. İletişim

Projelerde, iletişim önemli sorunlardan birisidir. XP bunu aşmaya çalışır. Ekip içi iletişime öncelik verir ve geliştirmeye çalışır.

12. Scrum

Scrum, büyük projeleri "sprint" adı verilen daha küçük projelere böler. İşi sprintlere ayırır ve her birini ayrı ayrı geliştirir. Scrum'da ekipler arası iletişim büyük önem arz eder, bu nedenle "Scrum Meetings" adı verilen günlük toplantılar yapılır. Düzenli geri bildirim ve planlamalarla hedefe ulaşmayı sağlar. Bu anlamda ihtiyaca yönelik ve esnek bir yapısı vardır. Müşteri ihtiyacına göre şekillendiği için müşterinin geri bildirimine göre yapılanmayı sağlar. 3 temel prensip üzerine kurulmuştur: Şeffaflık, Denetleme, Uyarlama.

Şeffaflık: Projenin ilerleyişi, sorunlar, gelişmeler herkes tarafından görülebilir olmalıdır.
Denetleme: Projenin ilerleyişi düzenli olarak kontrol edilir.
Uyarlama: Proje, yapılabilecek değişikliklere uyum sağlayabilmelidir.

12.1. Kavramlar

1) Product Backlog

Proje için gerekli olan gereksinimler listesidir. Proje sonunda "Ne üretilmek isteniyor?" sorusuna cevap aranır. Ürün sahibi tarafından müşteriden gereksinimler alınır,

öncelik sırasına göre sıralanır. Ürün sahibi, ihtiyaçlarının değişmesi durumunda product backlog'a ekleme veya çıkarma yapabilir. Böylece değişim, projenin her aşamasında projeye kolayca entegre edilebilir olur.

2) Product Backlog Item

Product backlog içindeki her bir gereksinime verilen isimdir.

3) Sprint

Proje sprint denilen küçük kısımlara ayrılır. Scrum içerisindeki tüm aktiviteler sprint içerisinde gerçekleşir. 1–2 haftalık süreçlerdir.

4) Sprint Backlog

Geliştirme takımı tarafından product backlog itemlar öncelik sırasına göre sprint içerisine alınırlar. Bir sprint boyunca yapılacak itemların listesi oluşturulur. İşlerin detaylı olarak zaman çizelgesi çıkarılır.

5) Scrum board

Bir sprint içerisinde yapılacak olan maddeler burada yönetilir. Yapılacak olan işler "TO DO" bölümüne alınır. Takım üyesi bu işe başladığında "IN PROGRESS" bölümüne getirilir. Bir iş, test için hazırsa "TO VERIFY" durumuna getirilir. İş, kontrol edildikten sonra "DONE" bölümüne getirilir. Scrum toplantılarında bu maddelerin durumlarına göre yerleri değiştirilebilir.

6) Burndown Chart

Yatay ekseninde sprintin günlerini, dikey ekseninde sprintte kalan işi gösteren grafiktir. Scrum'un temel ilkelerinden olan şeffaflığı sağlar.

12.2. Roller

1) Ürün Sahibi

Müşteri ile üretim ekibi arasındaki iletişimi kolaylaştırır. Projenin özelliklerini tanımlar. Proje hedeflerine uygun olarak product backlogu geliştirir. Sprint'i sonlandırabilir.

2) Scrum Yöneticisi (Scrum Master)

Scrum yönergelerini iletir, uyumu izleyip kontrol eder, toplantıları yönetir ve Scrum sürecindeki düzensizlikleri ele alır. Ekibin karşılaştığı zorluklardan onları kurtarmak için çabalar ve bu açıdan sorumluluk taşır.

3) Geliştirme Takımı

Bir sprint'e alınan bütün işleri tamamlayacak özelliklere sahip kişilerdir. Sprint backlogu oluştururlar. Kendi kendilerini yönetirler. İşin verilmesini beklemezler, işi kendileri alır ve geliştirirler. Kişilerin tek bir görevi yoktur, çapraz görev dağılımı yaparlar, herkes her şeyi yapabilir konumdadır. 5–9 kişi arasında değişir. Projenin geliştirilmesi ile ilgili sorumluluk geliştirme takımına aittir.

4) Chicken Roller

Scrum'ın işleyişinde aktif olarak yer almayan kişilerdir. Müşteriler, satıcılar gibi.

12.3. Toplantılar

1) Sprint Planning

Geliştirme ekibi, bu toplantıda product backlog'da listelenen gereksinimleri görevlere ayırır. Bu görevler, ekibin her üyesi tarafından kendi hızlarına aralarında paylaşılır. Bu toplantıya ürün sahibi, geliştirme ekibi ve scrum yöneticisi katılır. Her sprint, sprint sonunda ürün sahibine gösterilecek şekilde yazılım geliştirmeyi hedefleyerek belirlenir. 1-3 haftalık Sprintler üretilir.

2) Daily Scrum

Aynı mekanda aynı saatte yapılan günlük 15 dakikalık konuşmalardır. Üyeler davetiye beklemezler. Bu toplantı, gelecek günün planlarını yapmak için yapılır. Her takım üyesi şu sorulara yanıt verir: Dün ne yaptım, bugün ne yapacağım ve işimi yapmama engel olan herhangi bir sorun var mı? Bu şekilde, scrum yöneticisi ortaya çıkabilecek sorunları çözer. Tartışmanın sonunda, bu konuda yardımcı olabilecek biri varsa, bu takım üyesiyle iletişime geçebilirler. Her koşulda, bir daily scrum yapılır. Bir ekip üyesinin geç kalması veya gelmemesi toplantıyı etkilemez.

3) Sprint Review

Her sprintin sonunda yapılır. Sprint incelenir ve nihai çıktı değerlendirilir. Yazılımın ürün sahibinin gereksinimlerine uygun olarak oluşturulmasını sağlamak amaçtır. Hata varsa tespit edilir ve düzeltilir.

4) Sprint Retrospective

Bu konferansta sprintin iş kalitesi, doğru ve yanlışları değerlendirilir. Scrum takımının kendini daha iyi bir haline dönüştürmesi için bir fırsattır. "Neyi daha iyi yapabiliriz?", "Nasıl daha iyisini yapabiliriz." sorularına çözüm aranır. Bu aşamadan sonra, bir sonraki sprint planning toplantısı gerçekleştirilir ve her şey tekrar yaşanır.

Kaynaklar

https://ybsansiklopedi.com/wp-content/uploads/2015/08/Yazılım-Geliştirme-Modelleri-Yazılım-Yaşam-DöngüsüSDLCYBS.pdf
https://tr.wikipedia.org/wiki/Scrum
https://dergipark.org.tr/en/download/article-file/957302