Asım Kocamaz 220601151

YAZILIM YAŞAM DÖNGÜ MODELLERİ

Yazılım yaşam döngüleri yazılımları tasarlamak, geliştirmek ve test etmek amaçlı kullanılan süreçlerdir. Yazılımın nasıl geliştirileceği, sürdürülebileceğini açıklayan bir plandan oluşur. Bu döngü yazılımın aslında bir ürün olduğunu ve bu ürünün bir yaşam döngüsü olduğunu gösterir. Yazılım yaşam döngüleri müşterinin isteklerini karşılayacak şekilde gerekli maliyet tahminleri ile birlikte kaliteli ve işlevli yazılımlar üretmeyi amaçlar.

Yazılım yaşam döngülerinin 5 temel aşaması bulunmaktadır, bu 5 aşama şöyledir;

- Planlama: Yazılım yaşam döngüsünün ilk aşamasıdır, bu aşamada projenin neleri kapsadığı konuşulur ve projenin nasıl hayata geçirilebileceğinin hesapları yapılır.
- Analiz: Projenin ne kadar sürede tamamlanacağı, ne gibi riskler barındırdığı analiz edilir mevcut bilgiler doküman haline getirilir.
- Tasarım: Analizi yapılan proje tasarım aşamasına gelir ve burada projenin nasıl sonuçlandırılabileceği tartışılır, projenin nasıl devam edeceğine dair bir yol haritası çıkartılır ve problem daha basit hale getirilerek problemdeki önemli kısımlara dikkat çekilir.
- Test: Bu aşamada proje kodlanır ve test edilmeye başlanır, projenin istenilen amaca hizmet edip etmediği gözlenir, projede varsa kusurlar tespit edilir ve giderilir. En maliyetli ve zaman gerektiren adımdır.
- Bakım: Yazılım ürünü bütün testleri başarı ile geçtikten sonra, ürün piyasaya sürülür kullanıcılardan alınan geribildirimler ile gerekli geliştirme ve bakım süreci başlar.

Yazılım yaşam döngüsünde sürecin geliştirme aşamasında sürecin düzeni ve nasıl uygulanacağını ifade eden çok sayıda model bulunmaktadır. Bu modeller karışıklığı azaltır ve belli bir düzen sağlar.

Bu modellerden bazıları şunlardır; Şelale (Waterfall) modeli, V (V-shaped) modeli, Spiral model, SCRUM.

Şelale (Waterfall) Modeli: Yazılım geliştirme sürecinde analiz, tasarım, kodlama, test, entegrasyon gibi aşamalardan oluşur. En temel modeldir, bu adımların hepsi doğrusal olarak bir sonraki aşamaya doğru ilerler, geleneksel modellerin hepsi şelale yöntemiyle çalışır. Her aşama bir önceki aşamanın ürettiklerini kullanır ve kendi bünyesinde yaptığı değişikliklerle birlikte bir sonraki aşamaya aktarır.

Özellikleri şunlardır;

- Şelalenin her basamağındaki görev eksiksiz olarak yerine getirilmelidir bu bir sonraki aşamaya geçmenin şartıdır.
- Her aşamanın sonunda doküman oluşturulur.
- Yazılım süreci doğrusaldır, bir sonraki aşamaya geçebilmek için bir önceki aşamaların tamamlanmış olması gerekir.

- Kullanıcı katılımı başlangıç aşamasında mümkündür. Kullanıcının ihtiyaçları bu aşamada tespit edilir. Daha sonra gelen tasarım ve kodlama aşamalarında kullanıcı ve müşteri diyaloğu yoktur.

Bu özelliklerle birlikte bu modelin getirdiği pek çok problem de mevcuttur;

- Aşamaların birbirinden kesin olarak ayrı olması gerçekçi bir şey değildir projelerde aşamalar arasındaki sınırlar pek çok kez kaybolur.
- Teorik olarak safhalar birbirini kusursuz takip ederler fakat çalışma sırasında pek çok kez önceki aşamalara dönme ihtiyacı hissedilebilir.
- Model değişikliğe ve yeniliğe açık bir model değildir.
- Sistemin kullanılabilir hale gelmesi çok uzun zaman alabilir.
- Başlangıçta yapılan hataların fark edilmesi çok uzun zamanlar alabilir ve bu durum maliyeti arttırır.

Şelale modeli kullanışlı değildir çünkü müşteri talepleri ve istekleri her an değişebilir daha projenin başında müşterinin bütün taleplerinin detaylı analizini yapmak gerçekçi değildir ve olası bir değişiklikte fazlasıyla maliyet ve zaman kaybına yol açabilir.

V (V-shaped) modeli: Şelale modelinin daha gelişmiş halidir. Doğrusal bir şekilde ilerlemez, adımlar kodlama aşamasından sonra yukarıya doğru eğim alarak V şeklini oluşturur. Bu modelin Doğrulama aşamaları şunlardır;

- Gereksinim analizleri : Bu aşamada ilk evre doğrulama sürecidir. Sistem gereksinimleri kullanıcı ihtiyaçlarına göre toplanır. Bu evre ideal bir sistemin yapmak zorunda olduğu şeyleri kurgulamakla ilgilenir. Bu evrede yazılımın tasarımı ve uygulama yolu belirlenmez. Kullanıcı gereksinimleri dokümanları oluşturulur. Bu dokümanlar sistemin performansı ve güvenliği gibi kullanıcının bizlerden beklediği şeyleri tanımlar ve kullanıcıya sistemin anlatımını yapmak için iş analisti tarafından kullanılır. Kullanıcılar bu dokümanları inceler ve kabul testleri bu aşamada tasarlanır.
- Sistem tasarımı : Sistem tasarımı evresi sistem mühendislerinin analiz ettiği ve kullanıcı dokümanlarını çalışarak sistemin işleyişini anladıkları aşamadır. Olası durumları ve teknikleri kullanıcı ihtiyaçlarını gerçekleştirerek anlamaya çalışırlar. Eğer herhangi bir uyuşmazlık varsa kullanıcı bu konuda bilgilendirilir, çözüm bulunur doküman yeniden düzenlenir. Bu evrede iş senaryolarını, örnek pencereleri, daha iyi anlamak için raporlar tutulabilir. Diğer teknik dokümanlar da bu evrede üretilir.
- Mimari tasarım : Bilgisayar mimarisinin ve yazılım mimarisinin tasarım aşaması yüksek seviye tasarım olarak değiştirilebilir. Seçilen mimaride temel tipik olarak içereceği modüllerin listesi, her bir modülün özet fonksiyonelliğini, arayüz ilişkisini, bağımlılıklarını, veri tabanı tablolarını, mimari diyagramlarını, teknoloji detaylarını sunmalıdır. Entegrasyon test etme tasarımı özel bir aşama içinde gerçekleştirilir.
- Modül tasarımı : Modül tasarımı için düşük seviyeli tasarım demek doğru olacaktır. Sistem küçük modüllere veya birimlere ayrılarak her biri programcıya doğrudan kod yazabileceği şekilde aktarılır. Düşük seviye tasarım dokümanı veya program talimatnameleri sahte kod içindeki modülün ayrıntılı bir fonksiyonel mantığını taşıyacaktır. Veri tabanı tabloları, tüm elemanları ile birlikte tiplerini ve boyutlarını

içerir, tüm arayüz ayrıntıları karmaşık API referanslarıyla birlikte, tüm bağımlılık konuları, hata mesaj listesi, eksiksiz girdi ve çıktılar. Hepsi bu aşamada geliştirilir.

V modelinin içerisinde doğrulama aşamalarının her bir seviyesine karşılık gelen geçerli kılma aşamaları vardır. V modeldeki geçerli kılma aşamaları şunlardır;

- Birim test etme : V model içinde birim test planları modül tasarım aşaması boyunca geliştirilir. Bu planlar kodlardaki hataları gidermek için kullanılır. Birim testi kodun geri kalan kısımdan ayrı tutulduğunda düzgünce çalışabilen varlığı doğrular.
- Entegrasyon test etme : Entegrasyon test planları mimari tasarım aşaması boyunca geliştirilir. Bu testler birimler arasında haberleşebilen ve aynı zamanda birbirinden bağımsız şekilde test edilen ve oluşturulan birimleri doğrular. Test sonuçları müşteri ile paylaşılır.
- Sistem test etme : Sistem test planları sistem tasarım aşaması boyunca geliştirilir. Birim ve entegrasyon test planlarından farklıdır, sistem test planları müşterinin iş takımı tarafından birleştirilir. Sistem testleri uygulamanın gelişiminin beklentileri karşılayıp karşılamadığını kontrol eder. Tüm uygulama iletişim ve fonksiyonel olma açısından ayrıntılı şekilde test edilir.
- Kullanıcı kabul test etme : Kullanıcı kabul testi planları gereksinim analiz aşaması boyunca geliştirilir. Planlar kullanıcılar tarafından birleştirilir. Bu aşamada gerçekçi veriler üretim ortamının benzeri yaratılarak kullanıcı ortamında çalıştırılır. Bu aşama sitemin hazır olup olmadığını doğrular.

Spiral model : Spiral model risk yönetim için en çok destek sağlayan modellerden biridir. Şematik gösterimde birçok döngüsü bir spirale benzer ve bu döngülerin sayısı bilinmez. Bu spiralin her döngüsü yazılım geliştirme sürecinin bir aşamasına denk gelir. Ürünü geliştirmek için gereken aşama sayısı proje riskine bağlı olarak projenin geliştiricisi tarafından değiştirilebilir. Spiral modelde geliştiricinin görevi büyüktür. Spiral model riske bağlı bir modeldir bu yüzden geliştirme sürecini birden fazla kez yineleyerek risk yönetimi yapar. Spiral model aşamaları şunlardır.

- Planlama : Spiral modelin ilk aşamasıdır. Bu aşamada projenin kapsamı ve spiralin bir sonraki yinelenmesi için planlama yapılır.
- Risk analizi : Risk analizi aşamasında proje ile ilgili riskler belirlenir ve değerlendirilir
- Değerlendirme : Değerlendirme aşamasında ürünün yüksek kaliteli olup olmadığı veya amaca hizmet edip etmediği değerlendirilir.
- Planlama : Spiralin bir sonraki yinelemesi değerlendirme aşamasının sonuçlarına bağlı olarak yeni bir planlama aşamasından başlar.

Spiral model yazılım geliştirmeye daha esnek ve daha uygun bir alan sağladığından daha büyük ve kapsamlı projeler için tercih edilir. Ayrıca belirsiz ve risk düzeyi yüksek projeler için de uygundur. Spiralin herhangi bir noktadaki yarıçapı maliyeti temsil eder ve açısal boyut ise projedeki şimdiye kadar alınan ilerlemeyi anlatır. Spiral modelin her aşaması 4’ e bölünür bu 4 bölümün ayrı ayrı görevleri vardır. Bu bölümlerin görevleri şöyledir.

- Hedef belirleme ve alternatif çözümler belirleme : Müşterilerden gerekli bilgiler ve ihtiyaçlar toplanır, hedefler her aşamanın başında belirlenir. Daha sonra bu bölümde olası çözümler önerilir.
- Riskleri tanımlayın ve çözün : Bu bölümde mümkün olan tüm çözümler değerlendirilir. Daha sonra en iyi çözüm bulunur ve bu çözümün riskleri belirlenip bu riskleri gidermek için en iyi strateji yardımıyla çözülür. Bu bölümün sonunda prototip büyük oranda inşa edilmiştir.
- Ürünün bir sonraki sürümünü geliştirin : Bu bölümde belirlenen özellikler geliştirilir ve test yoluyla doğrulanır. Bu bölümün sonunda yazılımın sonraki sürümü hazırdır.
- Bir sonraki aşamayı gözden geçirin ve planlayın : Bu bölümde müşteriler yazılımın şimdiye kadarki geliştirilen kısmını değerlendirir ve sonun bir sonraki aşama için planlama yapılır.

Spiral model avantajlı olduğu kadar pek çok dezavantajı da mevcuttur. Bunlardan bazıları şunlardır; Spiral model diğer modellerden çok daha karışıktır, spiral modelin uygulanması pahalıdır ve bu yüzden küçük projeler için uygun değildir, projedeki aşamalar bilinmediği için zaman tahmini yapılamaz ve proje çok uzun sürebilir, spiral model birden çok yinelenme içerdiği için geliştiriciler için karmaşık olabilir, ve spiral model birden fazla değerlendirme içerdiği için zaman alıcı olabilir, spiral model için yüksek yatırımlar gerekebilir.

Scrum : Scrum ekiplerin kendi kendilerini organize etmek ve ortak bir hedef doğrultusunda çalışmak için kullandığı bir modeldir. Verimli proje için gereken bir dizi işlemi tanımlar. Yazılım ekipleri karmaşık sorunları uygun maliyetli ve sürdürülebilir şekilde çözmek için scrum modelini kullanırlar.Scrum modelinin proje başarısı için bazı ilkeleri şunlardır;

- Şeffaflık : Çalışanlar herkesin karşılaşabileceği sorunları bilir ve bunları bilerek çalışır farklı alanlarda çalışanlar düzenli toplantılarla olası anlaşmazlıkları hızlıca giderirler.
- Gözlem : Ekip üyeleri ilerlemelerini görmek için sık sık değerlendirme toplantıları ile gözlemlenirler bu toplantılardan elde edilen bilgilerle projenin geleceği planlanır. Bu sayede projeler daha verimli şekilde ilerleyebilir.
- Adaptasyon : Ekip üyeleri müşteri ihtiyaçları değişirse görevler arasında öncelik sırasını değiştirebilir.

Proje ekipleri için ise scrum ilkeleri farklıdır. Proje ekipleri için 5 adet scrum ilkesi bulunur;

- Bağlılık : Ekip üyeleri zamana dayalı görevlere bağlıdır ve en iyi çzöüm için sürekli gelişime odaklanırlar.
- Cesaret : Ekip üyeleri açık ve zorlayıcı sorular sorarlar ve en iyi çözüm için şeffaf tartışmalar yaparlar.
- Odaklanma : Ekip üyeleri kısıtlı zaman içeren görevler üzerinde çalışırlar,her görev sonrası yaptıklarını sunarlar.
- Açıklık : Ekip üyeleri her türlü yeniliğe ve fikre açık olmalıdır.
- Saygı : Ekip üyeleri birbirlerine ve liderlerine saygı duyarlar ve bu saygı karşılıklıdır.

Scrum öğrenmesi kolay fakat uzmanlaşması çok zor bir modeldir Scrum'ın özü, sprint adı verilen bir sınırlı süre içinde müşteri değeri sunan ve kendi kendini organize edebilen bir

ekiptir. Scrum, her bir Sprint ile ilişkili yapıt, rol ve etkinlikleri tanımlar. Bir scrum ekibi üç role ihtiyaç duyar bunlar ürün sahibi, scrum lideri ve geliştirme ekibidir. Ürün sahibi geliştirme ekibinin işletmeye en fazla değeri sunduğundan emin olmaya odaklanır. Müşterilerin değişen ihtiyaçlarını anlar ve onları bir sıraya koyar. Scrum liderleri ekiplerinin en üst üyesidir, ekibin verimliliği liderin sorumluluğundadır. Scrum süreçlerini iyileştirmek ve teslimatı gerçekleştirmek için ekiplere ve ürün sahiplerine koçluk sağlarlar. Scrum geliştirme ekibi test ediciler, tasarımcılar, kullanıcı deneyimi uzmanları, operasyon mühendisleri ve geliştiricilerden oluşur. Ekip üyeleri farklı becerilere sahiptir ve birbirlerini sürekli geliştirirler. Scrum etkinlikleri scrum ekibinin düzenli olarak yaptıkları şeylerdir. Sprint planlamada ekip bir sonraki sprintte tamamlanacak iş konusunda tahminde bulunur, üyeler ölçülebilir hedefler belirlerler ve planlama toplantısı sonun her bir ekip üyesi ürünün parçasının sprintten nasıl teslim edileceğini öğrenmiş olur. Sprint, Scrum ekibinin bir ürün parçasını bitirmek için birlikte çalıştığı gerçek süredir. Bir sprint genelde iki hafta sürer fakat projeden projeye bu süre farklılık gösterebilir. İş ne kadar karmaşıksa sprintlerin o kadar kısa olması gerekir. Günlük Scrum, ekip üyelerinin katılıp günü planladığı kısa bir toplantıdır. Üyeler, tamamlanan işleri bildirir ve Sprint hedeflerine ulaşmada yaşanan zorlukları dile getirir. Sprint sonunda, ekip bir araya gelerek tamamlanan işleri değerlendirmek ve paydaşlara göstermek için bilgilendirici bir oturum gerçekleştirir. Ekip, Sprint sırasında işe yarayan ve yaramayan noktaları belgelemek ve tartışmak üzere bir araya gelir. Ortaya çıkan fikirler, gelecekteki Sprintleri iyileştirmek için kullanılır.

KAYNAKÇA

https://aws.amazon.com/tr/what-is/scrum/

https://www.geeksforgeeks.org/software-engineering-spiral-model/

https://tr.wikipedia.org/wiki/V-Model_(Yaz%C4%B1l%C4%B1m_geli%C5%9Ftirme)

http://www.aspmvcnet.com/tr/m/yazilim-muhendisligi/selale-yontemi-waterfall-modeli.html

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

https://medium.com/@denizkilinc/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-temel-a%C5%9Famalar%C4%B1-software-development-life-cycle-core-processes-197a4b503696