**Batuhan Birkan 190601046**

Yazılım Yaşam Döngüsü Nedir?

Yazılım yaşam döngüsü, kaliteli ve gereksinimleri karşılayan bir ürün ortaya koymak için takip edilen sistematik aşamalar bütünüdür. Yazılım yaşam döngüsünde asıl amaç son ürünün, müşterinin ihtiyaçlarına cevap veren ve kaliteli bir ürün olmasına dayalıdır. Sistemin geliştirilmesi önceden belirlenmiş olan zaman, emek ve maliyet gibi hedeflerin gerçekleştirilmesi amacıyla mümkündür. Yazılım yaşam döngüsü yazılımın planı, yazılımın inşaa edilmesi ve bakımı gibi aşamaların detaylı planlanmasına dayanır. Aslında yazılım yaşam döngüsünün her aşaması kendi içerisinde de aşamalardan oluşur.

Yazılım Yaşam Döngüsünü Neden Bu Kadar Önemli?

1. Yazılım yaşam döngüsü bize projenin başlangıç ve bitiş tarihleri arasında uygun bir şekilde kronolojik plan yapabilmemize olanak sağlar ve öngörülebilir birkaç ipucu sunar.
2. Projede çok kullanılacak işlemleri ve fonksiyonları öngörmemizi sağlar ve biz de önceden bu fonksiyonlara göre uygun frameworkler geliştirebiliriz.
3. Projenin takibi ve kontrol mekanizması için oldukça gereklidir.
4. Projenin mümkün olduğunca berrak olmasını sağlayarak diğer developerlar arasındaki iletişimi kolaylaştırır böylece sürece de zaman katkısı sağlar.
5. Genel olarak projenin hızını arttırır.
6. Müşterilerle olan iletişimi kolaylaştırarak ilişkilerimizi daha etkin bir şekilde yönetmemizi sağlar.
7. Projenin olası risklerini azaltarak beklenmeyenlere karşı da hazırlıklı olmamızı sağlar.

Yazılım Yaşam Döngüsü Aşamaları Nelerdir?

1. Gerekli gereksinimlerin ve analizlerin derlenmesi
2. Projenin yapılabilirliği konusunda çalışma yapılması
3. Tasarım
4. İnşaa
5. Test
6. Entegrasyon
7. Bakım

Aşama 1: Gerekli gereksinimlerin ve analizlerin derlenmesi

Yazılım yaşam döngüsünün ilk aşaması gereksinimlerin analizidir. Süreç takımın daha tecrübeli kişileri tarafından yürütülmektedir. Kaliteli ürün için uygun gereksinimlerin belirlenmesi ve analiz edilmesi olası risklerin tespit edilmesi bu aşamada planlanmalıdır. Bu aşamayla beraber projenin yapısı, olması gereken hali daha da berrak bir hale gelir. Gereksinimlerin eksiksiz bir şekilde toplanması projenin istenmeyen sürprizlerle karşılaşmasını engeller ve projenin zamanında teslim edilmesi için gereken kronolojik planlamayı sağlar.

Aşama 2: Projenin uygulanabilirliği konusunda çalışma yapılması

Gereksinimlerin belirlenmesinin ardından bir sonraki aşama fizibilite çalışmasıdır.

* Projenin maliyetini hesaplar
* Yasalara aykırı herhangi bir unsurun olup olmayabileceği konusunu inceler
* Müşterinin istediği şekilde projenin çalışabileceği konusunu sorgular
* Teknik açıdan yeterliliği sorgular
* Projenin bitiş tarihini tahminen belirler

Aşama 3: Tasarım

Bu aşamada sistemin ve yazılımın gereksinimlerine cevap verecek olan uygun tasarımın ayrıntılı bir şekilde dökümante edilmesi beklenir. Böylece proje mimarisi genel hatlarıyla ortaya çıkmış olur. Bı aşamada 2 tip tasarım dökümnaını geliştirilir. Bu dökümanlar ve içerikleri:

1. High-Level Tasarım
   1. Her modülün ismi ve ne işe yaradığı konusunda bilgiler
   2. Her modülin fonksiyonal özellikleri
   3. Modüller arasındaki bağlantılar ve özellikleri
   4. Veritabanı ve veritabanındaki tableların isimlendirilmesi
   5. Mimarinin ayrıntıları ve kullanılacak teknolojiler
2. Low-Level Tasarım
   1. Modüllerin fonksiyonlarının mantıksal olarak incelenmesi
   2. Veritabanı tableları ve bununla beraber boyutları ve türleri
   3. Arayüzün ayrıntılı tasarımı
   4. Hata mesajlarının listelenmesi
   5. Her modülün input ve outputlarının belirlenmesi

Aşama 4: İnşaa (kodlama)

Sistemin tasarlanmasının ardından bir sonraki aşama kodlama aşamasıdır. Bu aşamada takım tasarım aşamasında belirlenen teknolojileri ve programlama dillerini kullanarak sistemi kodlamaya başlarlar. Kodlama aşamasında zaten ana başlıklar, birimlere ve modüllere ayrılarak yazılımcılara dağıtılmıştır. Genelde bu aşama yazılım yaşam döngüsünün en uzun aşamasını oluşturur. Bu aşamada yazılımcılar önceden belirlenen kodlama briefingleri takip ederler. Ayrıca bu aşamada kullanmaları gereken bir takım IDE’ler, derleyiciler, debuggerler vardır.

Aşama 5: Test

Yazılımın kodlanmasının bitirilmesinin ardından, yazılım test aşamasına taşınır. Test takımı sistemdeki fonksiyonları ayrıntılı bir şekilde test etmeye başlar. Yazılımın, müşterinin gereksinimlerini karşılayıp karşılamadığının ve sistemin düzgün bir şekilde çalıştığının doğrulanması için bu aşama önemlidir. Test takımı yazılımdaki bugları ve errorları tespit ederek bunları kodlama aşamasını gerçekleştiren takıma bildirir. Bunun üzerine diğer developerlar yazılımdaki bugları fixleyerek tekrar test edilmek üzere test takımına geri gönderir. Bu aşama yazılım gereksinimleri karşılayana kadar, düzgün bir şekilde çalışana kadar kendi içinde tekrar eder. Amaç bug bulunmayan stabil sürüme ulaşmaktır.

Aşama 6: Entegrasyon

Test aşaması başarıyla bitirilip buglar ve hatalar çözüldükten sonra artık sistemin entegre (install) edilmesi beklenir. Proje yöneticisinin geri bildirimleri temel alınarak son sürüm çıkarılır ve deploy edilir. Eğer deployment esnasında olası hatalar çıkarsa bunlar raporlanır.

Aşama 7: Bakım

Sistem entegre edildikten sonra ve müşteri sistemi kullanmaya başladıktan sonra ilgilenilmesi gereken 3 şey daha vardır:

1. Bugların giderilmesi: test edilemeyen senaryolar olacağından dolayı müşteri tarafından raporlanan bugların tekrardan giderilmesi gerekmektedir.
2. Yükseltme: yazılımın sürümünü yükseltmeye devam etmek.
3. Geliştirme: Gereksinimlere göre var olan sisteme yeni özelliklerin eklenmesi.

Yazılım Yaşam Döngüsü Modelleri

1. Waterfall Modeli (Şelale)

Waterfall modeli otoritelerce kabul gören ve en popüler modeldir. Bu modelde yazılım geliştirme süreci aşamalara bölünür. Her aşamanın sonucu bir sonraki aşama için bir bilgi olarak kullanılır. Az önceki bahsettiğimiz aşamalar bu modelde kullanılır. Bu modelin olayı bir akış halinde aşamaların birbirini takip etmesi ve aşamların görevleri bitince bir daha önceki aşamalara dönüş yapılmamasıdır. Geriye dönüş yapılmadığından da benzetme olarak Waterfall ismini almıştır.

Waterfall Modeli Özellikleri

1. Gereksinimler ayrıntılı ve anlaşılır bir şekilde dökümante edilmiştir.
2. Ürün gayet açık ve anlaşılırdır.
3. Teknoloji anlaşılmıştır ve statiktir.
4. Gereksinimlerde belirsizlik yaşanmaz.
5. Ürünün oluşturulması için gerekli kaynaklar ve bilgi elde edilmiştir.

Waterfall Modeli Avantajları

1. Kullanımı kolay ve basittir
2. Yönetimi kolaydır.
3. Aşamalar belirlenen zamanlara göre gerçekleşmiş olur.
4. Gereksinimlerin açık olduğu küçük projelerde kullanılabilir.
5. Hedef belirlemek daha kolaydır.
6. İşlemler ve sonuçlar ayrıntılı şekilde dökümante edilir.

Waterfall Modeli Dezavantajları

1. Son aşamaya kadar elde görülebilecek bir ürün ortaya çıkmaz.
2. Yüksek risk ve belirsizlik içerebilir.
3. Daha kompleks ve obje yönelimli projeler için iyi bir seçim değildir.
4. Uzun vadeli ve sürekli işleyen sistemler için yeterli bir model değil.
5. Gereksinimlerin değişme riskine karşı önlemli bir model değildir. Surpizlere karşı dayanıksızdır.
6. Değişen gereksinimlere uyum sağlayamaz.
7. Incremental Model

Incremental model tasarım, entegrasyon ve test aşamalarının az az olacak şekilde artımsal şekilde ve eş zamanlı olarak devam ettirilerek projenin tam olarak bitmesiyle son bulur. Bu model Waterfall modelindeki aşamaların elde bir protatip bulunması suretiyle yinelenmesi felsefesine dayanır. Bu modelde ürün önceden tasarımı yapılmış parçalara ayrılarak sürekli geliştirilir. Her parça ayrı şekilde tamamlandığı zaman müşteriye teslim edilir.

Incremental Modeli Avantajları

1. Çalışan ürün hızlıca ve erkenden elde edilir.
2. Gereksinimlerin değişebilme ihtimaline karşı daha esnek bir modeldir.
3. Küçük tekrarlamalar sayesinde test ve debug işlemleri daha da kolaylaşır.
4. Risk yönetimi daha kolaydır.
5. Her tekrar kolayca yönetilebilen dönüm noktalarına dönüşür.

Incremental Modeli Dezavantajları

1. Her aşamanın tekrarları kesindir ve kendisini tekrar etmez.
2. Bütün gereksinimler önceden elde edilmediğinden dolayı sistem mimarisine dair problemler ortaya çıkabilir.
3. V-Modeli

V modeli, yazılım yaşam döngüsündeki belirli aşamaların V şeklinde dizayn edilmesi ile ortaya çıkan bir modeldir. Ayrıca V modeli Waterfall modelinin geliştirilmiş halidir. Bu model disiplinel bir modeldir ve bir aşama bitmeden diğer aşama kesinlikle başlamaz. V modelinde birbirlerine paralele 2 çizgi çizilir. Sol taraftaki çizgide çeşitli tasarım aşamları yer alır sağ taraftaki çizgide ise test aşamaları yer alır. Ortadaki boşlukta da inşaa aşamaları vardır. V modelinin özelilikleri Waterfall Modeliyle hemen hemen aynıdır sadece aşamaların gerçekleştirilme şekli farklılık gösterir.

V-Modeli Avantajları

1. Yüksek disilpin içeren bir modeldir ve aşamalar tek zamanda tamamlanır.
2. Küçük projelerde temiz çalışır ve gereksinimler gayet açıktır.
3. Kullanımı ve anlaşılması kolaydır.

V Modeli Dezavantajları

1. Yüksek risk ve belirsizlik taşır.
2. Kompleks ve büyük projeler için uygun değildir.
3. Uzun vadeli projeler için uygun bir seçim değildir.
4. Test aşamasına gelen modülün geri dönüp fonksiyonlarının tekrar değiştirilmesi zordur.

4. Agile Model

Agile model incremental ve iterative modellerinin kombine edilmiş halidir agile modelde ayrı olarak müşterinin tatmin edilmesi ve işlemlerin aşamalara adapte olmasına odaklanır. Agile modelinde ürün küçük incermental yapılara ayrılır. Bu yapılar incremental modeldeki gibi arttırımsal tekrarlarla sağlanır. Her tekrar yaklaşık 1-3 hafta sürer. Agile modelinde her proje farklı şekilde ele alınır. Sık sık çalışabilen yazılımlar müşteriye sunulur ve müşteriyle sürekli bir iletişim halinde olunmalıdır. Agile modeli değişen gereksinimlere ayak uydurmayı amaçlar. Sadelik ve basitlik esastır. Elde her zaman bir çalışan bir protatip bulunur. Agile modeli 2001 yılında çıkarılar Agile Manifestosunda ilkelere ayrılmıştır. Bu ilkeler bireysel çalışma, her zmaan elde çalışan bir yazılımın bulundurulmaı, müşteri ilişkileri ve değişen gereksinimlere hızlı tepki verebilmek.

Agile Model Avantajları

1. Yazılım geliştirme sürecine gerçekçi bir yaklaşım sunar.
2. Bireysel çalışmayı ve takım çalışmasını kombin eder.
3. Sürekli bir geliştirme süreci vardır.
4. Kaynak gereksinimleri minimumdur.
5. Değişen veya değişmeyen gereksinimlere karşı kullanılabilir.
6. Sürekli dinamik projelerde kullanılması uygundur.
7. Planlama azdır.
8. Yönetim kolaydır.
9. Yazılım takımına esneklik sağlar.

Agile Model Dezavantajları

1. Müşteri ilişkileri çok hassastır bu nedenle eğer müşteri anlaşılmazsa takım yanlış yöne kayabilir.
2. Bireysel çalışmaya dayandığından dökümasyon az üretilir.
3. Teknolojilerin transferinde yeni takım üyeleri kolay uyum sağlayamayabilir.

5. Spiral Model

Spiral modelinde her aşamada bir hedef belirlenir ve her aşama müşteri değerlendirmesi sonucunda biter. Yazılım takımı küçük bir gereksinim listesiyle başlar ve her gereksinim için ayrı aşamalar oluşturulur. Yazılım takımı her yeni spirale girildiğinde yeni yeni fonksiyonlar ekleyerek projenin bitmesini sağlar. Plan, risk analizi, kodlama ve değerlendirme aşamalarından oluşur. Değerlendirme aşamasında müşteri büyük rol oynar. Müşteri değerlendirme aşamasında yazılımı değerlendirir.Spiral modeli büyük projelerde kullanılır. Sürekli yeni versiyonların çıkarılması gereken zamanlarda işe yarar. Risk ve maliyet değerlendirmesi büyük rol oynar. Riskli projelerde kullanılır.

Spiral Modeli Avantajları

1. Sonraki aşamalarda yeni fonksiyon ve eklemeler yapılabilir.
2. Maliyet belirleme protatip sürekli değiştiğinden kolaydır.
3. Sürekli ve tekrarlanan geliştirme süreci risk yönetimini kolaylaştırır.
4. Geliştirme süreci hızlıdır ve hızlı şekilde yeni sürümler çıkarılır.
5. Müşteri geribildirimi sürekli sağlanır.

Spiral Modeli Dezavantajları

1. Kronolojik planlama ve maliyet aksayabilir.
2. Büyük projelerde risk yönetimi oldukça önemlidir.
3. Uygun işleyebilmesi için modele kesinlikle paralel hareket edilmeli.
4. Sonraki aşamalarda dökümanatasyon daha da fazlalaşır.
5. Küçük projeler için uygun değildir.

Scrum Metodu

Müşteri ile olan iletişimin sağlıklı olmaması, sürekli değişen gereksinimler, değişen gereksinimlere uygun yeni modellerin projeye kolaylıkla entegre edilememesi gibi nedenlerden dolayı projelerde başarısızlık görülmektedir. Scrum metodu bu soruna bir çözüm getirir. Scrum kompleks yazılım süreçlerinin yönetimi için kullanılır. Tekrara dayanan bir metoddur ve bütünü parçalara ayırır. Scrum metodunun ilkeleri arasında: şeffaflık, projenin denetlenmesi ve dinamik gereksinimlere ayak uydurabilme gelir. Scrum metodunda çeşitli roller vardır. Bu roller:

1. Product Owner: Yazılım takımıyla müşteri arasındaki iletişimi sağlar.

2. Scrum Master: Takımı kurallara ve pratiklere karşı adapte eden kişidir.

3. Geliştirme Takımı: Çapraz görev dağıtımı yaparak bir tekrara alınan bütün işleri tamamlayan kişilerdir. İşi beklemezler işi alırlar.

Sprint Toplantıları:

1.Sprint Planlaması:Gereksinimlerin belirlenmesinin ardından geliştirme takımına proje parçalara ayrılarak küçük parçalar halinde dağıtılır. Developer takımı bu küçük taskları kendi kabiliyetlerine uygun olanları seçerler. Bu toplantıya prdouct owner, scrum master ve developer takımı katılır.

2. Günlük Scrum: Her gün aynı saatte ve aynı yerde ayak üstü yapılan15 dakikalık toplantılardır. Üyeler davet edilmeyi beklemezler Bu toplantılar günü planlamak için yapılır. Her takım üyesi kendi öz değerlendirmesini bu toplantıda sunar.

Sprint Değerlendirme: Her sprint sonunda yapılır. Yapılan sprint gözden geçirilir ortaya çıkan ürün değerlendirilir.