SDLC

1.slayt

Bugün bugün SDLC konusunda beraber olacağız ve diğer session da Devops konusuna değineceğiz.

Bu konunun devamı olarak yani yazılım geliştirme konusunda, 2 session Agile, ve son olarakta Jira derslerini göreceksiniz.

Bu dersten amacımız ne:

HTML, CSS, SQL gibi teknik konular size kodlamayı öğretir.

Bir uygulama nasıl kodlanır, ona nasıl şekil verilir işin teknik kısmını bu derslerle öğreneceksiniz.

Ancak işin bir de planlama ve proje kısmı var.

SDLC dersleri ile sizlere daha yukarıdan bir bakış açısı ile;

Yazılım geliştirilirken proje yönetimi nasıl olur işin management yönü nasıl yapılır sizlere genel fikir vermeyi planlıyoruz.

2.slayt

Arkadaşlar bugünkü dersimiz SDLC dedik yani Yazılım Geliştirme Yaşam Döngüsü.

Dersimiz Daha çok tanımların bulunduğu teorik ağırlıklı konuları kapsıyor.

SDLC konusunda google da arama yaptığınızda özellikle ingilizce çok fazla kaynak olduğunu görebilirsiniz.

Ancak burada her konuyu vermek yerine temel konuları size anlatmaya çalışacağım. Önemli kısımların altını çizerek tekrar edip genel resmi sizlere göstermeyi planlıyorum.

Amacımız sizler için bir temel atmak, bunun üzerine kat çıkıp bina inşa etmek sizlerin gayretleri ile olacak.

Bu dersi Türkçe anlatırken biraz da komik bi durum oluşuyor. “Plaza türkçesi” tabirini duydunuz mu?

\*\*\* youtube plaza türkçesi videosu aç\*\*\*

Birazdan anlatacağım derste de plaza türkçesi kullanacağız. Mesela; SDLC’de en önemli aşama Design Phase’dir. Bu phase logical design’ın functional desing’a dönüştüğü aşamadır. Gibi cümleler

IT sektöründe literatür ingilizce olduğu için bu jargona sizi alıştırmak için ders türkçe de olsa ingilizce ile karışık plaza türkçesiyle işleyeceğiz.

3.slayt

Dedik ki;

“SDLC, yazılım geliştirme için bir proje yöntemidir”

Software Development Life Cycle Yazılım Geliştirme Yaşam Döngüsü

Yazılımın geliştirilmesi esnasındaki yaşam döngüsüdür.

Bunu bir insan gibi ya da başka bir eşya gibi düşünürseniz daha kolay edersiniz.

Bir ihtiyaç var, bu ihtiyaç tanımlanıyor, daha sonra bu ihtiyacın tanımlanmadan sahip olunma aşamasına ve sahip olunduktan sonraki hayatına dair neler olduğunu kapsayan terim SDLC dir.

Şimdi lütfen buraya dikkat edin

Yazılım geliştirme unutun.

Bir masa ve sandalye şirketinin yöneticisi olduğunuzu düşünün. Bir müşteri sizi ziyaret etti ve masa istiyorum dedi.

En başta ilk tepkiniz ne olurdu?

Müşteriniz bir masa istiyor ve siz bu masanın üretim sürecini planlamak zorundasınız?

İlk ne yaparsınız?

Cevaplarınızı class chatten yazabilirsiniz.

Öncelikle soru sormaya başlarız.

- Bu masa ne için?

Yemek masası mı yoksa çalışma masası mı?

Aynı anda kaç kişi oturabilir? 2-4-8 kişi?

Neden yapılmıştır? tahta veya plastik olabilir

Böylece sorular sormaya başlıyorsunuz.

Durumu analiz ediyoruz

Çünkü istenen şeyi tam olarak anlamamız gerekiyor.

Bizden istenenleri ortaya çıkarmamız lazım.

- İsterler - Gereksinimleri nelerdir)

ikinci adım?

Bu gereksinimlere göre uygun tabloyu tasarlamanız gerekir.

Bu tasarıma göre işçiler masayı üretecekler.

Tasarımdan sonra hangi adım gelir?

Üçüncü adım, belirlenen masanın imalatı olan üretimdir.

Üretimden sonra ne yapmalıyız? Direkt müşteriye teslim mi yoksa???

Dördüncü aşama;

Evet!!!, ürününüzün kalitesini kontrol etmelisiniz.

ve sonra (beşinci aşama)

masayı müşteriye teslim edebilirsiniz.

Ama bitti mi?

İyi bir firma iseniz ürüne de sahip çıkmalısınız.

İşte bakım geliyor. (altıncı aşama)

Yazılım ürün yönetimini anlamayı kolaylaştırmak için bu benzetmeyi yaptım.

SDLC'nin de neredeyse aynı olduğunu göreceksiniz.

4. slayt

Şimdi dersimize biraz ısınalım, Peardeck’ten SDLC konusunda aklınızda ne varsa onu yazmanızı istiyorum.

SDLC hakkında ne biliyorsun?

Herkes yazsın lütfen yanlış yazarım diye düşünmesin, SDLC denince aklınıza ne geliyor yazalım.

Açılımı veya türkçesi nedir den ziyade SDLC denince aklınıza neler geliyor onu yazalım

En kısa şekli ile, tek cümle ile açıklayacak olursak SDLC “yazılım geliştirme” için uygulanan bir proje yönetim metodudur diyebiliriz. Birazdan konunun ayrıntılarına değineceğiz.

5. slayt

SDLC konusu neden önemli, şu an pek çoğunuz eski kariyerlerinizi geride bırakıp yeni bir alanda kariyer yapmaya doğru ilk adımlarınızı atıyorsunuz.

İnşallah ileride gireceğiniz işlerde yazılım projelerinde görev aldığınızda takip edilecek bir süreç var.

Bu süreçler firmanın yapısına, projenin büyüklüğüne göre değişiklik gösterse de temel dinamikleri aynı.

Bu kursta göreceğimiz SDLC, Agile, scrum, kanban, jira gibi konular ile, işte sizin gelecekte çalışacağınız işlerinizde takip edeceğiniz süreçleri, günlük çalışma şeklinizi, diğer takım arkadaşlarınız ile nasıl bir etkileşim içinde olacağınızı, en temel şekilde ortaya koymaya çalışacağız.

Teori olarak ne işleyeceğiz

* SDLC nedir, SDLC nedir bir cümleyle ifade edebiliyoruz ama bunu biraz daha detaylandırmamız lazım, sdlc sadece bir cümlelik bir tanım olsaydı insanlar bunun üzerine akademik araştırmalar yapmazdı, çeşit çeşit yöntemler geliştirmezlerdi. Bakalım insanlar neden buna bu kadar vakit ayırmışlar, önem vermişler onu göreceğiz.
* Safalar nelerdir, hangi feyzlerden oluşur onu göreceğiz
* SDLC modellerini göreceğiz
* Waterfall ( votırfol ) (Şelale) modelini biraz daha detaylı inceleyeceğiz.

SDLC nedir diye daha detaylı baktığımızda, şu yol simgesinden, resminden aklınızda şu kalsın. Nihayi ürüne (kullanıcının ihtiyaçlarını bir ürüne çevireceğiz ya ) en kısa zamanda, en az maliyetle, en yüksek kalitede ulaşmak için izlenecek yol arkadaşlar.

Biraz daha detaylandırdığımızda ( tahtaya yaz)

* En kısa zaman
* En az maliyet
* En yüksek kalite

Bunlar önemli, bunlardan herhangi birini yapamazsam müşteriye ihtiyacı olan ürünü versem bile iş işten geçmiş olabilir, müşterinin bütçesini aşmış olabilirim, müşterinin kalite ihtiyaçlarını karşılamamış olabilirim.

Bu yansıya baktığımızda, farklı yaklaşımlarda safaların isimlerinin farklı verildiğini göreceğiz. Yansılarda da farklı örnekler var, internette de sizin daha farklı yapacağınız araştırmalarda da bunun farklı örneklerini göreceksiniz.

İsimlere takılmıyoruz, bu dönemde yapılması gereken işlemlere dikkat ediyoruz sadece.

SDLC ürüne giderken bazı şeyler tanımlar, bu tanımların en önemlisi detaylı bir plandır.

SDLC ye ne kadar bağlı kalırsak geliştirme esnasında bizim devolopment hızımız artarken kaynak kullanımız azalacak. Bu da ne demek proje yönetiminde risklerin azalması demek arkadaşlar. Riskler azaldığı her durumda proje daha emin adımlarla ve daha düşük maliyetle başarıya doğru ulaşacaktır.

Tabi bunun için farklı yöntemlerde seçebiliriz, bunu birazdan göreceğiz, bize, şirketimize, kullanıcı ihtiyaçlarına hangisi uygunsa o metodla çalışmalarımıza devam edeceğiz.

Peki projenin karmaşıklığı arttıkça ne olmuş zaman içerisinde, eskiden yapılan programlarda sdlc ye gerek yoktu. Şimdi sistemlere, programlara ilk baktığımızda çok karmaşık, çok farklı teknolojiler bir arada kullanılıyor, bir sistemi yapmak için, html, css, java script, php, cango… Python, veri tabanı, bunların hepsini aynı anda kullanmanız gerekiyor, küçük küçük parçalarla.

İşte SDLC ile biz bu komplex liklerin, karmaşıklıkların üstesinden gelmiş oluyoruz.

Örneğin; burada ifade edilen döngüye baktığımızda birinci adım

* Analiz adımı ( Reguirement Analysis) , yani kullanıcı ihtiyaçlarının belirlenmesi. Sonra kullanıcıların girdilerini ve görüşlerini almış.
* Takiben bunlar ifade edilmiş, onaya sunulmuş (alıntı ve onaylamak)
* Mokap----Görsel olarak hazırlanmış ama uygulamaya konmamış ani bir web sitesi düşünün görsellik var tasarım var ama butonlar çalışmıyor onu görüyorsunuz.
* Kodlama
* Test aşaması
* Ürünün teslim edilmesi

Şu 3 noktada kullanıcı girdileri alabilir şeklinde bu saykılda ifade edilmiş.

\*\*

Bir başka gösterime de bakarsak arkadaşlar, burada da yine benzer bir yaklaşım ama biraz daha farklı isimlendirmeler, farklı safa sayılarıyla da ifade edildiğini görebilirsiniz.

Burada ne var;

* Requirement Analysis---analiz kısmı
* Desing----dizayn
* İmplementation ------ coding
* Testing
* Evolution ---geliştirme olarak adlandırılmış

Bir başka yaklaşımda ise burada rikayrımınt analisis den önce kikof, yani projenin başlatılması safası konmuş. Bu tip gösterimlerde olmuş.

Project Kickoff---Bu aşamada ne olabilir, sözleşme ihtiyaçlarını önceden tanımlamak gerekebilir, projenin çerçevesinin çizilmesi, yaklaşık maliyetinin belirlenmesi bu aşamalarda yapılmak istenebilir.

Bu hem proje yönetimi hem de kullanıcı ihtiyaçlarına yönelik olabilir. Kullanıcı ihtiyaçları sadece yazılıma yönelik değil, bu sürece yönelik ihtiyaçlarda olabilir.

* Rukayrımınt gethering (ge’dirin)-----Gereksinimleri toplama
* Analiz (e’nalisis)
* Dizayn (dezing)
* Development (divelıpmınt) --- implementeşin--- cooding-----bunlar yazılımın geliştirilmesi için kullanılan terimlerdi.
* Test
* Deployment---dağıtım (yayılma)
* Opereşın and meıntınıs (apıreşin end meintınıns) --- operasyon ve bakım------yani burada yazılım geliştirici ekibin işi bitmemiş arkadaşlar. Yani işletme ve bakımını da devam edecek şekilde döngüler göstermiş.

SDLC

Software Development Life Cycle ın özelliklerine biraz daha detaylarına bakmak istersek, zaten sizin kafanızda da ufak ufak yapılması gerekenler şekilleniyor olacak.

* Bir yazılım projesi için izlenecek sistematik süreç
* Yazılım oluşturmanın ve geliştirmenin yapılandırılmış yolu

SDLC , strakçır, yani yapılandırılmış, belli bir yapısı , sistemi olan işlemlerdir. Software Development Life Cycle ı rastgele, kafama göre bir gün A dan başlayıp D den bitirip, bir gün D den başlayıp B de bitirecek şekilde yapmamam gerekiyor. Bunun sistematik ve yapılandırılmış bir süreç olması gerekir.

Aynı şekilde yüksek kalitede bir süreç çalıştırmam gerekir. Bu direk ürünün kalitesini etkileyecektir.

Düşük maliyetli bir yazılıma ulaşmam gerekir.

Sınırsız kaynak verdikten sonra her insan her şeyi yapar. Önemli olan kısıtlı kaynaklarla maliyette bunlardan bir tanesi, bunu yapabilmek ve Mümkün olan en kısa üretim zamanında biz bunu üretmemiz gerekiyor.

SDLC nin amacı nedir ?

Yani ne olduğuna baktık, rastgele bir sistem değil. Şimdi bunu niye ben bunu kullanıyorum, bunu kullanmasam ne olur ?

Bunun amacı, gelişkin, geliştirilebilir, üst kalitede bir yazılım üretmek, birinci amaç bu ve bu amacı gerçekleştirirken de ürettiğim yazılımın kullanıcının istek ve gereksinimlerini , arzularını karşılaması veya üzerinde olması gerekiyor.

Kullanıcı ihtiyaçlarını değiştirebiliriz, bundan yanlış bir anlam çıkmasın, yani biz kullanıcının isteklerinin üstünde bir yazılım üretebiliriz.

SDLC nin neden önemli olduğunun üzerinde biraz daha duralım, SDLC development sürecimizi hızlandırır.

Kendinizi bir yönetici veya şirketin bir üst düzey çalışanı olarak düşünürseniz, ( hepiniz zamanla öyle yerlerde çalışacaksınız) hiç kimse projesinin gerekenden bir gün daha fazla sürmesini istemez.

Bir diğeri, müşterilerinizle ilişkilerinizi güçlendirir. Şimdi strakçır ve sistematik bir proses demiştik ya, siz bu prosesin içerisine müşteriyi ta en baştan dahil etmiş oluyorsunuz. Kendinizi müşteri yerine koyun, size hiçbir şey sormadan yazılım yapan firmayla mı çalışmak istersiniz, sizi daha en başından isteklerinizi, arzularınızı maliyet karşılayabilme gücünüzü, her şeyi dikkate alıp bunu sistematik bir şekilde dahil edip sizle çalışan ve bunu ürüne dönüştüren, takiben aralarda sizinle toplantılar yaparak kontrol eden biriyle mi çalışmak istersiniz.

Yönetici olarak tekrar düşündüğünüzde bir projede en görmek istemeyeceğiniz şeyler risklerdir arkadaşlar (risk yönetimi ap ayrı bir konu ) , siz bir projede ne kadar risk görürseniz bunların her birini ayrı ayrı yönetme ihtiyacı duyacaksınız, bazılarını belki yönetemeyip zamana bırakmanız gerekecek, bunlar hep sizin projenizin başarısız olma ihtimalini arttıran sebepler olarak karşınıza çıkacak.

Bunun engellenmesi de yine uygun bir SDLC metodu kullanmak önemli olacaktır.

SDLC bize projeyle ilgili tahminler yapma ve bunları bir takvime bağlama imkânı sunduğu için yine yazılım süreci için önemlidir.

Biz en başından itibaren yazılım geliştirme sürecinde bir plana ve takvime bağladığımız için kontrol etmesi de kolay olacaktır. Takip etmesi de kolay kontrol etmesi de kolay olacak.

Siz ürün şu gün şu tarihte ürün şu seviyede olmalı dediğiniz yerde veya ben kullanıcı ihtiyaçlarınızı analiz etmeyi bitirmeliyim dediğiniz yerde bakacağınız yer plan olmalı, bu iş bitti mi bitmedi mi. Veya html tarafı bitti, css tarafı bitmedi şeklinde kontrol mekanizması kolaylaşıyor.

Ve bütün bunların da etkisi maliyet , bizim bu saydıklarımızın ve risklerin de takviminde bir maliyeti var. Biz bunları baştan iyi yönetirsek maliyeti de karşılamış oluyoruz.

Biz burada teoriden biraz çıkıp geldiğimiz yere kadar olan konularla ilgili örnek göstermek istiyorum.

Bu linkler başlangıç için güzel noktalar.

<https://pmbasics101.com/software-development-life-cycle/>

Yaklaşık olarak ne anlam ifade ettiğini bildikten sonra gerisini araştırmak için biraz vakit ayıracağız.

* Satış Öncesi Aşama
* Proje Başlangıç ​​Aşaması
* Konsept Geliştirme Aşaması-- Gayrimenkul geliştirme projelerinde satın alma gücünü zorlamayacak konseptlerin mutlaka en verimli ve ilgi çekici şekillerde sunulması tavsiye edilmektedir.
* Prototipleme Aşaması (Genellikle, projenin fizibilitesini sağlamak için bir Faz Kapısıdır.) Prototip, bir ürünü piyasaya sürmeden önce ürünün neye benzeyeceğini, ürünün ne yapacağını ve ürünün nasıl işlediğini gösteren gerçek bir ürün simülasyonudur.
* Planlama aşaması
* Gereksinim Tanımlama Aşaması
* Kullanıcı Arayüzü Tasarım Aşaması
* Geliştirme aşaması
* Entegrasyon
* Test Aşaması
* Dağıtım Aşaması
* Hiper Bakım Aşaması
* Bakım Aşaması
* Devir, Kapanış, Destek Faz(lar)ı

<https://pmbasics101.com/project-charter-example/>

Her şeyi ezbere bilmenize gerek yok, zaten bir işe girdiğinizde size ne yapmanız gerektiği söylenecek, zamanla yönetici konumuna da geldiğinizde hizmet içi eğitimlerle de bu bilgiler verilecek. Siz ne olduğunu bilin yeterli bunlar.

SDLC sürecinde neler yapıyorum, evet kullanıcının ihtiyaçlarını aldım. Bunları bir dökümana dönüşütürüyorum, bu dökümanlar daha sonra dizayna dönüşüyor, arada kullanıcıyla tekrar etkileşime girip projeyi kontrol ediyorum, bunları da ufak ufak geliştirmiş oluyoruz.

Yüksek kaliteli yazılım denince aklınıza ne geliyor? (BLA BLA BLA)

iyi bir tasarım,

işlevsellik,

özelliklerin uygunluğu ve fazlalığı,

iyi bir performansa sahip platform ve konfigürasyonlar

son kullanıcının kullanımına uygun yazılımlar (kötü bir ara yüzü kullanıcı beğenmez- kullanamaz-)

SDLC'nin Aşamaları

SDLC'nin kaç fazı vardır?

Yansılarımıza devam edersek, SDLC nin kaliteyi arttırdığını ve aynı zamanda da maliyeti düşürdüğünü söylemiştik.

Bunu peki nasıl yapıyoruz ?

Burada bir örnek var, bu örneği tek doğru olarak kabul etmiyorsunuz, önemli olan neler yapıldığı buradaki işlemler. Bunların hiç birinin başlıklarını şu aşamada ezberlemenize gerek yok.

Burada farklı bir safalandırma kullanılmış yine

Formeşin

Rukayrımınt planing

Dizayn

Konstrakt -----yerine ne kullanabilirdim----development, coding, implementeşin

Test

Prodak riliz------diploymınt, instoleyşin( yazılımın yüklenmesi)

Post implementeşin----implementeşindan sonrası diye ayırmış.

Projek inişeşın ın altında kabaca projenin geliştirilme tahminleri yapılmış,

biz bunu rukayrımınt definişin altına da koyabiliriz.

Burda ne geliyor;

projek menecimint plan, yani proje yönetim dökümanı.

Bunun altında, fankşınıl rikayrımıntları tanımlayacağım demiş, fonksiyonel ihtiyaçlar yani benim sitem sörç imkanı sunmalı, bu bir fankşınıl rikuayrımınttır değil mi.

Teknikıl rikuayrımınt a baktığımızda, benim sitem sörç yaparken anında sonuç vermeli Google gibi (ajax kullanarak) yada olduğu gibi kalsın normal arama yapsın diyebilir (pahalı diye)

Bu rurikuayrımıntların gözden geçirilmesi ve onaylanması

Derken iş planına çevrilmesi diye ikinci safayı tanımlamışlar.

3. safada (dizayn ) intörnıl/extörnıl (dahili ve harici) ele alınması,

Bunun gözden geçirilmesi, ve detaylı bir plana dönüştürülmesi.

4. Konstrakt, development, koding safhasında ise, önce bir protetif ( ilk örnek) yapmışlar sonra codu riviv ( gözden geçirme, yeniden inceleme) etmişler.

Traning e buradan başlamış, ne yapmış buradan itibaren kullanıcıyı alıp ben eğitmeye başlayacağım demiş.

Dökümenteşin a nereden başlamış, dizayndan itibaren, kullanıcını dökümantasyonundan, sizin yaptığınız yazılımı alıp karşısında oturup çalışacak insanın kullanacağı dökümantasyon.

5. Daha sonra test i yapacağım demiş, sistem testi, test özeti.

6. Prodak riliz, Opereşinil ekseptıns yapıcaz, sistemin gerçek şartlarda çalışabilirliğinin kabulünü yapacağız ve bunu dökümandan oluşturucağız demiş.

Proje bittikten sonra da neler yapılabileceği ile ilgili alt adımları tanımlamış.

Html de site yaparken bunları düşünebilirsiniz….. (düşündünüz mü ?)

SDLC kaç safhadan oluşur?

SDLC konusunda bir kaynak taraması yaparsanız

safhalarının farklı kaynaklarda farklı sayılarda belirtildiğini görebilirsiniz.

Genellikle 4 ten 8 e kadar aşamalandırıldığını görmek mümkündür hatta 10 olarak yazan bile var.

Ancak Yoğun olarak bizim preclassta olduğu gibi 6 aşama olarak belirtilmektedir

Rikuayrımınt, fizibility, dizayn, coding, testing, instol deploy, meintenins

Bu yeterli olacaktır, önüne bir şey ler eklenebilir belki.

Evet SDLC den şu kadarını anlayabildik mi;

SDLC bir metodoloji, yazılım geliştirirken kullandığımız sistematik bir problem çözme yöntemi, bunun safhaları var, safhaları değişebilir, farklı isimler kullanılabilir, önemli olan hangi safhada ne iş yapıldığını aklımda tutmam, buraya kadar anlattığımızın ana mantığı bu. Sorusu olan yada konuya kadar eklemek istediğiniz var mı ?

Eğitim in kalitesi arttıkça işe gösterdiğimiz özen arttıkça her şeyi plan ve program dahilinde yapma ihtiyacı doğmuştur.

* Her bir safha bir öncekinin ihtiyaçlarını karşılamalı ve bir sonraki safhaya da gerekli ürünü vermeli.
* Rikuayrımınts (ikinci madde), gereksinimler tasarıma dönüşür, tasarım devolopmınta dönüşür, ve developmınt teste, test sonucu da müşteriye verilecek ürüne dönüşür ve o safhanın girdisi olur.
* Her şey ne ile başlıyor arkadaşlar, rikuayrımıntla başlar.

\*\*

Aslında nasıl safhalandırılmış olursa olsun tüm phase’ler benzer faaliyetlerin gruplandırılmış halidir.

Kimisi isterler ve analiz safhasını tek bir safha olarak ele alırken kimisi bunu iki ayrı safha olarak ele almış. Veya ismini farklı vermiş.

O yüzden internette farklı kaynaklarda görebileceğiniz değişik safhalandırmalar kafanızı karıştırmasın

4 safhada da gösterilse 8 safhada da gösterilse nihayetinde süreç içinde yapılan faaliyetler hemen hemen aynıdır

SDLC de Her safha kendisinden önceki safhanın çıktıları üzerine inşa edilir.

Analiz safhasında ortaya çıkan gereksinimler, design safhasında tasarıma dönüşür

Bu tasarı da neler olur;

hangi programlama dili kullanılacak, hangi database gerekli, ihtiyaçları karşılayacak donanımlar nedir

Bunlar kesin olarak belirlenir.

development veya coding safhası bu temel üzerine bina edilir.

Testing aşaması coding aşamasından gelen çıktıları kullanır, değerlendirme ona göre yapılır.

Bu değerlendirmeler neticesinde programın yayımlanmasına müşteriye teslim edilmesine karar verilir ya da hataların düzeltilmesinden sonra teslim edilmesi planlanır.

En kritik aşama nedir? En öncelikli aşama hangisidir

Requirement safhası önde

Evet bu aşamada müşteri ne istiyor ortaya konur. Burada yapılacak bir hata, elde edilecek çıktının baştan hatalı olmasına neden olur.

SRS ne anlama geliyor ( pre-claslarda olması gerekiyor )

Software Requirements Specification Document

Yazılım gereksinimlerini (tanımlama ) detaylandırma dökümanı

1. Rikuayrımınt safhasında, kullanıcının ihtiyaçlarının alındığını söylemiştik.

Requirement Phase ihtiyaçların, gereksinimlerin tespit edilmesinin, sistematik olarak ortaya konmasının sağlandığı SDLC'nin ilk aşamasıdır.

Bu aşamaya planlama, tanımlama veya analiz safhasıda denebilir.

Bu aşamada müşteri, ürün veya yazılımla ilgili gereksinimleri, şartnameleri, beklentileri ve diğer özel gereksinimleri belirtir.

İşletme yöneticisi veya proje yöneticisi (project manager)(product owner) tüm bu bilgileri ve ön koşulları toplar.

Bu safhanın sonucunda herkes projeyle ilgili , yazılımla ilgili çok daha net bir resim görmeye başlar.

Safhanın sonucu olarak da ne ortaya çıkar ; Software Requirements Specification Document ortaya çıkar arkadaşlar.

Küçük bir ekip de olabilir, daha büyük bir ekip olarak da bu safha gerçekleştirilebilir.

Bu safhada yazılımcı kullanıcının gerçekten ne istediğini daha iyi anlar, aynı zamanda kullanıcı da yazılımcının neler ortaya koyabileceğini, istediklerinin bazılarının kendisine pahalıya patlayabileceğini, aklına gelen bazı isteklerin evet bu da olmalı diye fikir alışverişlerinin olabileceği bir safha olur.

Yazılım sistemini geliştirmek için, istenen yazılım hakkında net bir fikre sahip olmamız gerekir.

Bunun yolu da müşterilerle sürekli iletişim halinde olmaktan geçiyor.

Bu aşamada yapılacak iş, oluşturulacak ürünün fonksiyonel ve teknik gereksinimlerini belirlemektir.

Karşı tarafı tanı, ihtiyaçlarını anla, iletişimi geliştir.

Bu belgenin adı neydi arkadaşlar?

Software Requirements Specification Document kısaca SRS’tir.

Türkçeye çevirecek olursak “Yazılım gereksinimleri şartnamesi” diyebiliriz.

Bu döküman aslında bizim proje ile ilgili neler istendiğinin bir listesi.

Çeklist gibi düşünebilirsiniz. Proje bittiğinde biz bunları istemiştik neler yapılmış bir kontrol listesi gibi buradan bakılabilir.

\*\*\*

Bu aşamada müşteri, ürün veya yazılımla ilgili gereksinimleri, özellikleri, beklentileri ve diğer özel ihtiyaçları belirtir. Business owner veya project manager tüm bu bilgileri ve ön koşulları toplar.

Yazılım sistemini geliştirmek için, istenen ürünü, yazılımı net bir şekilde anlamalıyız. Bunu başarmak için, tüm gereksinimleri toplamak üzere müşterilerle sürekli iletişim kurmamız gerekir.

Her şey iki tarafça açıkça anlaşılıp kesinleştikten sonra dokümantasyon yapılmalıdır.

Neden? çünkü bütün proje bu dökümana göre ilerleyecek, tasarımcılar buna göre tasarlayacak, yazılımcılar da ona göre yazacak.

SDLC sample SRS document template

Burada farklı bir örnek, burada ne var, çaptırlara bölmüş, introdakşın, skop, sözlük, user rikuayrımınt defineşin…. Diye farklı dökümanlara bölmüş.

Software rikuayrımınt dokümanını ayrıntılı inceleyecek olursak,

Buradaki örneğe baktığımızda 3 temel başlıkta toplamayı tercih etmiş,

* İntradakşın, giriş, amaç, kabulleri, kim okuyacak bu dökümanı, tavsiyeler, projenin kapsamı, referanslar konmuş.
* Ovırıl diskripşın, ürünün perspektifini ortaya koymuş, ürünün özelliklerini, kullanıcı sınıflarını, hangi çevrede çalışacağını ( kutuplarda çalışacağınız istenirse buna göre donanım olmalı ) , beni bağlayan şeyler ne, kullanıcı dökümantasyonu nasıl olacak, kabullerin ve bağlılıkların olduğu yer.
* Spesifik rikuayrımınts ------ Özelliğin önceliği ne , etki tepki sırası ne, fonksiyonel gereksinimi ne, hangi program diğer hangi programa ihtiyaç duyar, performans gereksinimleri her bir özelliği biraz daha detaylandırıyoruz. Ex, farklı donanımlarla etkileşimli olabilir, bir iletişim sistemleri olabilir, kullanıcı arayüzleri olabilir.

Safhaları konmuş.

Kullanıcı ihtiyaçlarını ben detaylandırıyorum, teknik olarak ne yapmam gerekiyorsa onları çeviriyorum, transleyt ediyorum ( hangi donanımı kullanmam gerekecek, hangi programlar kullanılarak yapılacak ) ve bunların hepsini belli bir sistematik içerisinde dökümanlaştırıyorum.

<https://medium.com/ad%C4%B1m-ad%C4%B1m-srs-haz%C4%B1rlamak/alex-slobozhan-a-ait-https-freshcodeit-com-sitesinde-yan%C4%B1nlanan-yaz%C4%B1n%C4%B1n-t%C3%BCrk%C3%A7eye-terc%C3%BCmesidir-15f5395c2a68>

1. Fizibility : İkisi aslında iç içe olan diğer bir safhamız nedir, fizibility yani uygunluk, uygulanabilirlik , yapılabililik olarak çevirebiliriz bu fizibility i. Sizce bir projede fizibility neler üzerinden yapılabilir. Hangi başlıklar üzerinden yazılımın fizibıl olup olmadığını düşünebilirim.

Teknik olarak fizibıl mı ----yazılım ve donanım açısından uygun mu kullanıma

Opereşınıl fizibiity ----- bizim yazdığımız yazılımı kullanıcı anlayabiliyor mu, kaabiliyeti var mı

Skecıl ----zaman açısından

Finansal -----maddi açıdan uygun mu

Legal mi değil mi

Dolayısıyla 5 başlıkta biz bu fizibility analisizi yapıyoruz arkadaşlar. Bunlardan her hangi birinde uygulanabilirlik sıkıntısı görüyorsam kronolojiye bu andan itibaren bu şekilde devam etmemem lazım. Ne yapmam gerekir, kullanıcıya dönmem gerekir, veya bunlara çözüm bulabiliyorsam çözüm bulmam gerekir, bulunamıyorsa olay farklı türde gelişecektir.

1. Bir sonraki safha nedir, Dizayn (tasarım )

Farklı seviyelerde ele alınabilir. Önce sistemin bir genel tasarımını yaparsınız.

Prezenteşin tiır, burada sadece kullanıcının neler göreceği ve ekranda neler olacağı ile ilgili bilgi verirsiniz.

Aplikeyşin seviyesinde bir dizayn yaptığınızda ise artık serverlarınızın, programlarınızın nasıl çalışacağı, kullanıcıdan nasıl veri alıp bunları nasıl geri vereceği, bunları tasarlarsınız.

Data tiır, veri seviyesinde tasarlama yapabilirsiniz, database serverlarınızı tasarlayabiirsiniz.

Tasarım esnasında farklı toollarda kullanabilirsiniz. Tool nedir, yazılım esnasında kullandığım araç, bu bir yazılım dili olabilir, veri tabanı yönetim sistemi olabilir, server olabilir (apaçi olabilir)

Neye ihtiyacım olabilir, veri tabanına ihtiyacım olabilir, veri tabanında ben sql server, mysql server, mongo db bunlardan herhangi birisini kullanabilirim.

Dizayn aşamasında amaç nedir, hedefler

<https://www.openxcell.com/blog/design-phase-in-sdlc/>

dizayn aşamasında sistemin genel halini çıkarmış oluyoruz.

Riquayrımınts bizim sorunumuzdur, problemimizdir, dizayn ise çözümümüzdür. Aklınızda kalması gereken en önemli şey bu.

“iyi bir yazılım” için iyi bir tasarıma ihtiyaç duyulmaktadır.

Demek ki tasarımı baştan iyi yapmazsanız ne kadar iyi kodlar yazarsanız yazın sisteminiz çökebilir.

1. Coding ----

Tasarımın nihayi olgunluğa ulaşmasıyla birlikte Kodlama aşaması başlar.

Diğer adıyla yazılım geliştirme aşaması, bu aşama Müşteriye teslim edilecek ürünü programlama aşamasıdır.

Daha önceden tasarım aşamasında karar verilen ortama uygun olarak, yine tasarım aşamasında verilen kararlar doğrultusunda projenin gerçekleştirilmesine başlanır.

bu safhada developerlar (geliştiriciler) kod yazarak sistemi oluşturuyor. Programlama dili hangisiyse, bir gurup Python da çalışırken diğer bir gurup html, javascript, c de java da çalışıyor olabilir. Burada görevlerde alt parçalara bölünür. Farklı farklı görevlendirmelerle coding safası başlar.

Bu genellikle en uzun safhadır. Neden kodlama yaptıkça bunu görmeye başladınız, bazen iki satırlık bir işlem günlerinizi alabilir.

İhtiyaca göre dil seçimi, tasarıma göre görev paylaşımı, kim hangi fonksiyonları yazacak, kim hangi modülü yazacak, kim hangi kütüphaneyi yazacak (laybery) , bunlar arasında görev paylaşımı yaparak kodlama bölümünü bitireceğiz.

\*\*\*Bu safaların odak noktası nedir

1. Yazılım tamamlandıktan sonra arkadaşlar, test ortamına alınır. Şehir içinde küçük bir internetiniz olabilir test edilir, gerçek bir server da internet ortamına koyup test yapabilirsiniz bu da olabilir, bu ayrı bir kurs tur arkadaşlar. Zaten araştırma yaparken de görmüşsünüzdür. Ayrıntılarına girmeye gerek yok, siz sadece test ekibi ne yapıyor, bu safha ne yapıyor onu bilmeniz size yetecektir.

Bu safhada ürünü alıyoruz, yazıldığı kadarıyla başlıyoruz kullanıcının ihtiyaçlarını karşılıyor mu diye.

Burada kullanıcı ihtiyaçlarını tek tek test dokümanına çevirip kullanıcı bunu istemiş yapıyor mu evet kullanıcı bunu istemiş yapıyor mu evet kullanıcı bunu istemiş yapıyor mu evet, şeklinde bir akışla gider. Ve farklı yöntemleri de deneyebilirsiniz.

Mesela bir entegrasyon söz konusudur, programları birim birim test edersiniz, entegrasyon testi yaparsınız, sistemin bütününün testi yapılabilir, olayı bitirir test ekibi ve install aşamasına geçilir.

Ürünün SRS’belgesinde tanımlanan kalite standartlarına erişene kadar,

üründeki hataların, tespit edildiği raporlandığı, düzeltildiği ve tekrar test edildiği aşamadır.

1. Deployment, bu safhada ise herhangi bir bag olmadığını, kullanıcının bütün ihtiyacını karşıladığını gördükten sonra instoll, deployment başlar.

Bu nedir arkadaşlar, eğer sistem Amazon üzerinde çalışacaksa o kodları alıp ilgili instance lere yüklenmesi, ayarlanmalarının yapılması (yok locale yapılacaksa ilgili serverlara yüklenmesi ile ilgili ayarlanmaların yapılması ) , bununla ilgili script lerin yazılması, bunlar uygun ortamlara diploy edilir.

Ancak ürün teslim edilmeden yapılacak son bir işlem daha vardır.

Kullanıcı kabul testi (KKT veya User Acceptance Test),

yeni geliştirilen yazılımın/ürünün piyasaya sürülmeden önce yapılması gereken son ve en önemli proje aşamalarından biridir.

Bundan dolayı KKT’ne, yazılım için köprüden önce son çıkış diyebiliriz.

* Genellikle uygulamayı kullanacak olan son kullanıcı test eder.(İlgili iş birimi).
* Mesela bir firmanın ihtiyaçlarına uygun olarak customer management / müşteri yönetim programı hazırladınız. O firmada hangi departman kullanacaksa o kişilerin test etmesine imkan verilir.
* Gereksinimlerin karşılandığını görülünce son kullanıcının onayını alıp teslim işlemine geçilir.

Buradan da onayı aldıktan sonra ürün teslim edilir.

Eğer bir web sitesi yaptıysak canlı bir web server a eklenir, eğer bir veri tabanı yaptıysak canlı bir veri tabanı sunucusuna yüklenerek instollar yapılabilir.

1. Meintenens a geldiğimizde bu da farklı farklı sürelere yayılabilir. Anlaşmamız şudur, ben sana bir aylığına meintenens veririm kullanıcı da bunu ister bir ay sonra bütün anlaşmamız biter, daha sonraki her şey ilave paradır. Veya öyle bir program yapmışsınız ki kullanıcı sizden 15 yıl 20 yıl sizden destek isteyebilir, siz de bunu fiyata yansıtıp sağlarsınız işinize geliyorsa. Bu süreçte çıkan bag lar, upgrate ler ve geliştirmeler meintenens safhasının konularıdır.

Elinizdeki telefonları düşünün milyonlarca insanlar programları yüklüyor güncellemeler gelebiliyor, hatalar çıkabiliyor, bu işin doğasında var, bu safhada da hatalar çıkabilir. Bunları düzeltmek de meintınıns safhasında düzeltilecek şeyler.

\*\*\*\*\*

SDLC'nin ne olduğundan ve aşamalarından detaylıca bahsettik, şimdi sizlere SDLC modelleri hakkında genel bir bilgi vereceğim.

Modeller

Her biri farklı yaklaşımlar gerektiren birçok farklı SDLC modeli var.

SDLC modelleri, bir yazılım geliştirme projesinde,

her aşamada gerçekleştirilen etkinliklerin birbirleriyle nasıl ilişkili olduğunu açıklar.

Bunlara geçmeden önce yansıdaki soruyu cevaplayalım lütfen

Yaygın SDLC modellerini listeleyin ???????

Biz bu modeller içinde geleneksel yöntemlerden Waterfall,

Modern yaklaşımlardan Agile ve DevOps modeli en bilinen modellerdir.

Bu derste Şelale (waterfall) modelini göreceğiz.

Modeller

Bunları intervievde sorduklarında ya da konuşulduğunda ne anlama geldiğini bilmeniz yeterli, en önemlileri bu üçü bunları da ayrıntılı göreceğiz zaten.

Aslında hangi Model olursa olsun SDLC’nin aşamaları takip ediliyor ancak fark uygulanış biçiminden kaynaklanıyor.

Ortada yapılması gereken bir proje varsa, proje yöneticisi veya şirketin kendisi nasıl bir proje yönetim modelini takip edeceğini belirler.

Günümüzde yazılım şirketlerinin genellikle Agile ve bunun içinde de scrum veya kanban modellerini kullandıkları görülmektedir arkadaşlar.

Bunları iki eksenli tablo şeklinde göstermemiz gerekirse

Sol taraftaki, sekuneşin, yani birbirini takip eden adımlarla giden demek.

Evoleyşınıry ise sürekli birbirini takip eden yapıda olanlar demek, onlar üst tarafta gözüküyor.

Formıl, çok katı kuralları olan, onun dışına çıkamadığımız resmi methotları gösterirken,

Sağ tarafa da gittikçe daha informıl yöntemler ortaya çıkıyor.

Amacımız da kaliteyi arttırmak.

SDLC, belli bir sistem ve süre içerisinde yazılım geliştirme sürecidir.

Modeller ilgili oldukları dönemin donanım ve teknolojileri çerçevesinde ortaya çıkmıştır arkadaşlar.

Bundan 25 yıl önce en iyi model şelale modeliydi.Şimdi Agile denilliyor.

Belki 5 yıl sonra yeni modellerden bahsediyor olacağız

Hangisi geleneksel SDLC modeli -------- wodırfoll

Bu modelleri agile (modern ) ve tredeşınıl (geleneksel ) olarak ikiye ayırdığımızda öne çıkan şeyler şunlar;

Öncelikle tredeşınıl, proses ve toollar kullanır, çok detaylı doküman ihtiyacı vardır bunların, anlaşma pazarlıkları sürer, çok sıkı bir plana dahildir.

Fakat agile metodolojisine baktığımızda ise, etkileşim önemlidir, bireyler arası etkileşim önemlidir. Burada amaç, çalışan bir yazılım üretmektir, Kullanıcıyla sürekli etkileşim halinde, gerekirse kullanıcının ihtiyaçlarını gözden geçirmesine fırsat verme ve bu değişikliklere de bizim uyum sağlamamız olarak ön plana çıkıyor.

Waterfall Model

Wodırfoll modele bakarsak, bunun tredeşınıl bir yöntem olduğunu biliyoruz, geleneksel olduğu için.

Peki bu ismi nereden gelmiş, wodırfol suyun belli bir yol izlemesine ve bunu sürekli yapmasından ileri geliyor. Su A---B---C geçmeden ulaşamaz. Aynı mantık.

Bu modelin anlaşılması ve kullanılması çok basittir. Modeli, yazılım geliştirme sürecinin aşamaları arasındaki geçişi bir şelalenin dökülmesi gibi düşünebiliriz.

Peki bunun hangi yıllarda çıkmış olabileceğinizi düşünürsünüz, 1970 ( pre clas)

Eski bir yöntem ama hala kullanılabiliyor.

Gereksinimleri iyi tanımlanmış ve belgelenmiş projeler için çok uygun bir modeldir. Bu yüzden gereksinimlerin önceden belirlendiği ve değişimin çok az olacağı projelerde kullanılır.

Dokümantasyonun bir süreç aşamasından ziyade üretimin her aşamasında bulunması ve yapılması gerektiğini öngörür, dolayısıyla bir nevi doküman odaklı diyebiliriz.

Daha önce de belirttiğim gibi waterfall modeli en eski ve kabul görmüş modellerden biridir.

Esasında ilk olarak imalat ve inşaat sektörü gibi faaliyet alanlarında kullanılmıştır.

Bilgisayar teknolojisindeki ilerlemeye paralel olarak o dönemki yazılım geliştirme süreçlerinde de kullanılmaya başlandı.

O dönem için bu yöntem yeterli denebilir.

Günümüzdeki gibi pek çok farklı yazılımın bulunmadığı, web veya mobil uygulamaların olmadığı, ürün tanımının sabit olduğu nisbeten küçük projelerin geliştirilmesinde yeterliydi.

Buradaki olay şu,

Rikuayrımınts analisis başlar biter dizayn aşamasına geçilir…… coding----testing-----meintenence

Ve bu döngü tamamlandıktan sonra geriye dönüş olabilir. Bu da birbirini takip eden bir yazılım geliştirme süreci olduğunu söylemiştik.

Doğrusal bir akışta birbirini takip eden aşamalardan oluşan ve bir işlem bir safha bitmeden diğerine geçilmeyen bir yazılım geliştirme metodolojisidir.

Proje kapsamı küçük ise bunu tercih edebiliriz.

Safhaları;

Analiz, ihtiyaçların tanımlanması

Süreç içerisinde yenilik ve değişiklik yapılamayacağı için bu safhada istekler çok net şekilde ortaya konmalıdır.

Bu safhanın çıktısı bir sonraki safhada kullanılacak olan SRS dökümanıdır.

Projenin detaylandırılması SRS Software Requirements Specification Document

Software dizayn dokümınt, SDD, proje planı

İmplemteşin ın diğer adı neydi hatırlayan var mı, coding, development (elinde kahve )

Testing

Opereşin, meintenens

* Peki kodlama wotırfol modelde en uzun zaman alan bölümdür. Evet
* Şelale modeli hakkında ne düşünüyorsunuz arkadaşlar eski moda mı? kullanılamaz mı?
* Wodırfol modelin avantajlarına ve dezavantajlarına baktığımızda, neden insanlar bunu kullanır, uygulaması kolaydır.

Toplantı serileriniz, arada almanız gereken kararlar yok, sırayla yapıyorsunuz her şeyi , bu bitti mi yenisine geç devam, bu bitti mi….. sırayla ilerliyorsunuz, takibi kolay, uygulaması kolay, böyle bir avantajı var wotırfol modelin.

Her geliştirme aşaması için son tarihler ile beraber sabit ve net bir program belirlenebilir.

Bilgisayar konusunda, proje yönetim konusunda çok gelişkin olmak zorunda da değilsiniz. Küçük bir ekiple, tecrübesi daha az bir ekiple de bunu yapabilirsiniz.

Toparlamak gerekirse;

* Basit ve anlaşılması kolay projeler için idealdir.
* Yönetimi kolaydır
* Aşamalar birer birer tamamlanır, belli bir sıra takip edilerek belirsizlik ve karmaşadan uzak kalınır. .
* Küçük projeler için iyi ve düzenli işleyen bir süreç yaratır.
* Açıkça tanımlanmış aşamalar bulunur
* iş akışını düzenlemek kolaydır.

Ama dezavantajlarına baktığımızda ise, aslında sıralı olması bir dezavantaj. Kodlamada bir tasarım hatası fark ettim ne yapacağım, her şey boşa gidecek.

En büyük dezavantajı ise bunun arkadaşlar, süre!!!!. Üst aşamalarda oluşabilecek hatalardan dolayı zaman kaybı fazladır.

Gereksinimlerin değişebileceği projeler için uygun değildir.

projede yenilik veya değişiklik yapmak çok zordur

Müşteri ürünü en sonunda gördüğünden yazılımın reddedilme riski vardır.

yani bu, yüksek miktarda risk ve belirsizlik olduğu anlamına gelir.

Risk ve belirsizlik çok yüksek.

Bu uygulamayla sonuca gitmeniz her zaman kolay olmayabilir, uygulaması kolay olabilir.

Wodırfol un safalarına baktığımız zaman SDLC nin özetini görmüş oluyoruz.

KAHOTT

1. SDLC nin açılımı nedir
2. SDLC'de kaç faz vardır?----hepsi
3. Aşağıdakilerden hangisi SDLC süreç aşamaları arasında DEĞİLDİR?
4. Wodıfol modelinde , tüm yazılım geliştirme süreci ayrı aşamalara bölünmüştür.
5. Wodıfol modeli aşağıdaki durumlardan hangisinde etkili bir şekilde kullanılabilir?
6. Aşağıdakilerden hangisi bir yazılım geliştirme modeli DEĞİLDİR?
7. Wodıfol SDLC'de Modern Bir Modeldir

KAHOOT SDLC 1

1. SDLC'nin son aşaması nedir?
2. SDLC'de kaç faz vardır?
3. Aşağıdakilerden hangisi SDLC süreç aşamaları arasında DEĞİLDİR?
4. Yazılım Geliştirme Yaşam Döngüsünün (SDLC) en kritik aşaması nedir?
5. Kodlama aşaması olarak da bilinen aşama hangisidir?
6. Waterfall yaklaşımında, tüm yazılım geliştirme süreci ayrı aşamalara bölünmüştür.
7. Aşağıdakilerden hangisi bir yazılım geliştirme modeli DEĞİLDİR?
8. SDLC'deki 1. adım nedir?
9. Waterfall, Agile And Spiral Tümü SDLC Modelleridir
10. Waterfall, geleneksel bir SDLC modelidir

İterative Model

Yinelemeli modelde, yinelemeli süreç, küçük bir yazılım gereksinimleri kümesinin basit bir uygulamasıyla başlar ve tamamlanana kadar gelişen sürümleri yinelemeli olarak geliştirir.

Sistem uygulandı ve kullanıma hazır.

Yinelemeli bir yaşam döngüsü modeli, gereksinimlerin tam olarak belirtilmesiyle başlamaya çalışmaz. Bunun yerine geliştirme, diğer gereksinimleri belirlemek için yazılımın yalnızca bir kısmını belirleyip uygulayarak başlar.

Bu süreç daha sonra tekrarlanarak modelin her yinelemesinin sonunda yazılımın yeni bir sürümü üretilir.

Her yinelemede tasarım değişiklikleri yapılır ve yeni işlevsellik sağlanır. yetenekleri eklenir. Bu yöntemin arkasındaki temel fikir, tekrarlanan döngüler (yinelemeli) ve bir seferde daha küçük bölümler (artımlı) yoluyla bir sistem geliştirmektir.

Modülün sonraki her sürümü yeni işlevler ekler önceki sürüme. İhtiyaca göre komple sistem hazır olana kadar süreç devam eder.

Bu modelin avantajı, geliştirmenin çok erken bir aşamasında sistemin çalışan bir modelinin bulunmasıdır; bu model, işlevsel veya tasarım kusurlarını bulmayı kolaylaştırır.

Sorunları geliştirmenin erken bir aşamasında bulmak, sınırlı bir bütçeyle düzeltici önlemlerin alınmasını sağlar.

Bu SDLC modelinin dezavantajı yalnızca büyük ve hacimli yazılım geliştirme projelerine uygulanabilmesidir. Bunun nedeni, küçük bir yazılım sistemini daha küçük, kullanışlı artışlara/modüllere bölmenin zor olmasıdır.

Spiral Model

Spiral model, risk analizine çok önem veren, yinelemeli geliştirme süreci modeli ile şelale modeli gibi sıralı doğrusal geliştirme modelinin bir kombinasyonudur.

Ürünün kademeli olarak piyasaya sürülmesine veya spiralin etrafındaki her yinelemede kademeli iyileştirme yapılmasına olanak tanır.

Spiral Model, herhangi bir ürünün doğal gelişim süreciyle (örneğin olgunlukla birlikte öğrenme) senkronize olduğundan ve aynı zamanda hem müşteri hem de geliştirme firmaları için minimum risk içerdiğinden yazılım endüstrisinde yaygın olarak kullanılmaktadır. Peki bu model nerede kullanılabilir;

* Bütçe kısıtlaması olduğunda ve risk değerlendirmesi önemli olduğunda
* Orta ve yüksek riskli projeler için
* Gereksinimlerin zamanla değişmesi nedeniyle ekonomik önceliklerdeki olası değişiklikler nedeniyle uzun vadeli proje taahhüdü
* Müşteri gereksinimlerinden emin değil ki bu genellikle böyledir
* Gereksinimler karmaşıktır ve netlik elde etmek için değerlendirme yapılması gerekir
* Yeterli müşteri geri bildirimi alabilmek için aşamalı olarak piyasaya sürülmesi gereken yeni ürün grubu
* Geliştirme döngüsü sırasında üründe önemli değişiklikler bekleniyor

V-Modeli

V modeli, işlemlerin V şeklinde sıralı bir şekilde yürütüldüğü SDLC modelidir. Doğrulama ve Doğrulama modeli olarak da bilinir.

V-Model, şelale modelinin bir uzantısıdır ve ilgili her geliştirme aşaması için bir test aşamasının ilişkilendirilmesine dayanır. Bu, geliştirme döngüsündeki her bir aşama için doğrudan ilişkili bir test aşamasının olduğu anlamına gelir. Bu oldukça disiplinli bir modeldir ve bir sonraki aşama ancak bir önceki aşama tamamlandıktan sonra başlar.

V-Model kapsamında, geliştirme aşamasının ilgili test aşaması paralel olarak planlanmaktadır. Yani 'V'nin bir tarafında Doğrulama aşamaları, diğer tarafında ise Doğrulama aşamaları var. Kodlama aşaması V-Modelinin iki tarafını birleştirir.

V-Model'in avantajı anlaşılmasının ve uygulanmasının çok kolay olmasıdır. Bu modelin basitliği aynı zamanda yönetimini de kolaylaştırır.

Dezavantajı ise modelin değişikliklere karşı esnek olmaması ve günümüzün dinamik dünyasında çok yaygın olan bir ihtiyaç değişikliği olması durumunda değişikliği yapmanın çok pahalı hale gelmesidir.

RAD Modeli

RAD (Hızlı Uygulama Geliştirme) modeli, hiçbir özel planlama gerektirmeden prototip oluşturmaya ve yinelemeli geliştirmeye dayanmaktadır.

Yazılımın yazılma süreci, ürünün geliştirilmesi için gerekli planlamayı içerir.

Hızlı Uygulama geliştirme, atölye çalışmaları veya odak grupları aracılığıyla müşteri gereksinimlerinin toplanmasına, yinelemeli konsept kullanılarak prototiplerin müşteri tarafından erken test edilmesine, mevcut prototiplerin yeniden kullanılmasına, sürekli entegrasyona ve hızlı teslimata odaklanır.

Söylediğim gibi Hızlı uygulama geliştirme (RAD), hızlı prototipleme lehine minimum planlama kullanan bir yazılım geliştirme metodolojisidir.

Detaylı bir ön planlamanın olmaması, değişikliklerin geliştirme sürecine dahil edilmesini kolaylaştırır.

RAD projeleri yinelemeli ve artımlı modeli takip eder ve geliştiriciler, etki alanı uzmanları, müşteri temsilcileri ve diğer BT kaynaklarından oluşan ve bileşenleri veya prototipleri üzerinde aşamalı olarak çalışan küçük ekiplere sahiptir. Bu modelin başarılı olabilmesi için en önemli husus geliştirilen prototiplerin tekrar kullanılabilir olmasını sağlamaktır.

Hızlı Uygulama Geliştirme (RAD), hızlı prototip sürümlerine ve yinelemelere öncelik veren bir tür çevik yazılım geliştirme metodolojisidir. Şelale yönteminden farklı olarak RAD, sıkı planlama ve gereksinimlerin kaydedilmesi yerine yazılım ve kullanıcı geri bildiriminin kullanımını vurgular

\*\*\*

Geleneksel SDLC, kodlama başlamadan önce gereksinim analizine ve toplamaya büyük önem veren katı bir süreç modelini takip eder. Müşteri üzerinde, proje başlamadan önce gereklilikleri imzalaması yönünde bir baskı oluşturuyor ve uzun süredir çalışan bir yapı bulunmadığından müşteri ürünü hissedemiyor.

Müşteri, yazılımı gerçekten gördükten sonra bazı değişikliklere ihtiyaç duyabilir, ancak değişiklik süreci oldukça katıdır ve geleneksel SDLC'de ürüne büyük değişiklikler eklemek mümkün olmayabilir.

RAD modeli, çalışma modellerinin müşteriye yinelemeli ve artımlı olarak sunulmasına odaklanır. Bu, müşteriye hızlı teslimat ve ürünün tüm geliştirme döngüsü boyunca müşterinin katılımıyla sonuçlanır ve gerçek kullanıcı gereksinimlerine uygun olmama riskini azaltır.

Kütüphane ye yazılım yapmakla veya yaptırmakla görevli olsam ne yapmalıyım, SDLC modeline göre nasıl yapmalıyım.

1. Requirement Analysis

* ~~Kütüphanenin cinsi~~
* Kitap temin yöntemi her bir kitap için belirlenmeli ve kayıt altına alınmalı ( Nitelikli koleksiyon burada olmalı )
* ~~Yıllık bütçe~~
* Güncellik ( kitapların güncel olması )- kütüphaneciye yeni çıkan kitapları tarayıp sisteme dahil edilmeli

Daha önceki yazarın yeni kitabı için bildirim

Aradığım gelince bana mesaj atsın

* Sistem aşağıdaki kabiliyetleri içeren bir web sitesi olmalı

Arama fonksiyonları olmalı

En çok okunan kitaplar sayfası olmalı

En çok okuyan okuyucular sayfası olmalı

Kullanıcı yönetimi olmalı

İki aşamalı doğrulama bilgileri tutmalı

Etkinlikler kısmı olmalı

Rezervasyon sistemi

Kullanıcı dostu bir sistem olmalı-personel için

Hangi kitap şu an nerede olmalı

Online kitap sorgulama

Kullanıcı bilgilerinin alınması

Kullanıcıların ayrı hesapları olmalı ve yetkilendirme yapılmalı

* ~~Online hizmet sunmalı~~
* ~~Nitelikli koleksiyon~~
* ~~İşletecek bir kütüphaneci~~ ( nihai ürün hazır olduğunda kurumun 5 personeli eğitilmeli )
* Kitabı veren ve alan için 1 hafta veya 3 gün kala mail ile haber vermeli
* Tüm veriler bir veri tabanında tutulmalı ( olmalı )

Sistem Cloud temelli vt ile çalışmalı

İçerik olarak kitaplar tasniflenmeli

~~Kitap girişi için belli personelin yetkilendirilmesi~~ ( kullanıcıların yetki düzeyleri olmalı ) ( kullanıcı tipi ve yetkileri müşteri tarafından verilecek )

Kitapların takibini tasnife göre kişiler görevlendirilmeli

İki aşamalı doğrulama bilgileri tutmalı

VT sistemi yedekli olmalı

Hangi kitap şu an nerede olmalı

Kullanıcı bilgilerinin alınması

Daha önceki yazarın yeni kitabı için bildirim

Yaş ve cinsiyet analizleri

Kullanıcıların ayrı hesapları olmalı ve yetkilendirme yapılmalı

Kitapların ad yazar vs tutulması

* Teslim alınırken hasarsız geri alınması
* Sistem kullanım geçmişine göre raporlar üretebilmeli

Kişi analizi yapıp yeni gelen kitaplardan bilgilendirme maili alma

Daha önceki yazarın yeni kitabı için bildirim

* Sistemiin günün belli saatlerinde çalışması-güvenlik
* Siber güvenlik
* Sistem IOS Android vs desteklemeli ( ıos 10.03 ve üzeri dest
* Teslimat öncesinde test yapılması ???
* Takımdaki her birisine ilgili ihtiyacı uzmanlık alanına göre anlatıp anladığına emin olunmalı ve aradaki koordinasyon sağlanmalı
* ~~En çok okunan kitapların listesi~~
* ~~Sayfada çocukları teşvik için ödül/başarı panosu~~
* ~~Memnuniyet formu~~
* ~~Süre uzatmak için talep formu.~~

1. Fisibility

ekonomik

Sistem Cloud temelli vt ile çalışmalı ---- 15000 $ tek seferde yıllık 500 dolar

kütüphaneciye yeni çıkan kitapları tarayıp sisteme dahil edilmeli ----100000$

legal

kullanıcı bilgileri kayıt izni

operation feasibility

evet

tecnical

evet

Schedule

24 ay

1. Desing

Kullanıcının rikuayrımıntını dizayna dönüştürüyoruz burada

* Bu modül kullanıcının istek ve şikayetlerini almak için kullanılır
* Ana menüden ulaşılır
* Veriler memnuniyet tablosuna kayıt edilir
* Form doldurulup kaydedildikten sonra yöneticiye mail olarak iletilir

Mesela hata mesajları ne olabilir

İsim girilmedi

Çok uzun yazıldı

Uygunsuz yazı dili kullanıldı

Yanlış doğrulama

4. Coding / İmplementation

Takımdaki her birisine ilgili ihtiyacı uzmanlık alanına göre anlatıp anladığına emin olunmalı ve aradaki koordinasyon sağlanmalı

Bu taleplere göre iş planı oluşturmak

Takımları oluştur

Sorumlulukları belirlee

Her bir takıma görev dağıt

Modül-fonk-kütüp ne gerekiyorsa yapılacak

VT ekibi ile Analiz ekibi sgünlük toplantılar yapmalı

Her modülün teslim tarihi belirlenmeli

5. Testing

Teslimat öncesinde test yapılması

UL test

Requirements test

Arama fonk @ in başına ve sonuna ne yazılacak hata veriyor

Yetki düzeyleri belirlendi tasarım dökümanına uygun çalışıyor

6. Deployment /install

Web server setup

Database setup

DB ile data giriş

7. Maintenance

* Onarım ve bakım

Bu kütüphane yazılımını ben hazırlasaydım sistem gereksinimleri tam olarak çıkaramazdım, etkileşim çok önemli.

İsteklerin ve ihtiyaçların ne kadar detaylı bir şekilde incelenmesi gerektiğini görmüş olduk, anlatınca havada kalıyor her şey.

Bu safha en kritik safhaydı, burada yaptığınız bir hata diğer aşamalarda da sıkıntıya sebebiyet verecektir.

Siz SDLC dersinden önce ne düşünürdünüz, şimdi neler değişti ?????

metin, web sitesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

SDLC nin olayı bu arkadaşlar, strakçırt ve sistematik bir şekilde yazılım geliştirme sürecini çalıştırmak

1. Requirement Analysis

* Bu bölümde yer alan ifadeler muğlak olmamalı
* Burada tanımladığım gereksinimler projeye ait olmalı
* Gereksinimlerin müşteriden gelmesi tercih sebebidir
* Gereksinimler genelden detaya doğru kırılmalı
* Gereksinimler alt sistemlere göre guruplandırılmalı

Kullanıcı için önemli olanları seçmeliyiz.