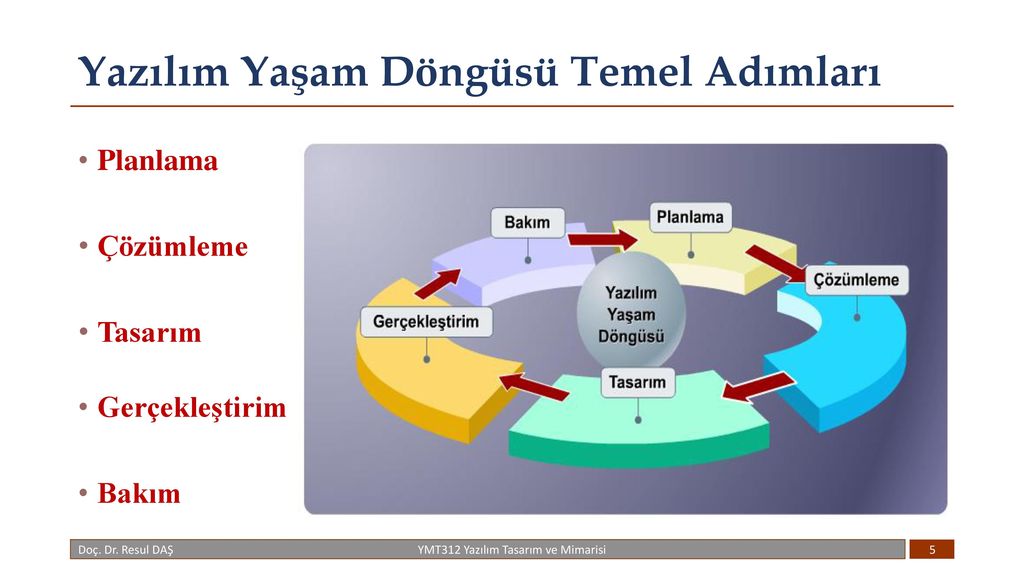
Yazılım yaşam döngüsü (“software development life cycle”, “SDLC”) : Yazılım bir üründür ve her ürünün de belirli bir yaşam süreci (döngüsü) vardır. Bu yazılımın, ihtiyaç durumundan gerçekleştirilmesi ve sonrasında müşterilerin kullanmasına kadar geçen tüm sürece yazılım yaşam döngüsü denir . Yazılım yaşam döngüsünün kendi içinde adımlardan (aşamalardan) oluşur . Toplamda 5 adım vardır ve bu adımlar sırasıyla ilk olarak planlama , ikinci olarak analiz , üçüncü olarak tasarım , dördüncü olarak gerçekleştirim (kodlama ve test) ve sonuncusu ise teslim ve bakımdır .



1.) Planlama : Yazılım yaşam döngüsünün ilk adımıdır . Bu aşamanın asıl amacı ilk olarak gereksinim analizi yapılır . Projenin planlaması ve görev dağılımı yapılır .

2.) Analiz : Yazılım yaşam döngüsünün ikinci adımıdır . Bu aşamanın asıl amacı mevcut yapıdaki işlerin ortaya çıkarılması ve doğru olarak anlaşıldığından emin olmaktır . Bu aşamada belirli gruplardan (iş analisti , yazılım mühendisi , sistem analisti , ürün yöneticisi gibi) yardım alınabilir hatta müşterilerle konuşarak projenin daha iyi hale getirilmesi sağlanabilir .

3.) Tasarım : Yazılım yaşam döngüsünün üçüncü adımıdır . Planlama ve analiz aşamasında belirlenen ihtiyaçlara çözüm olacak yazılımın ilk temelleri atılır . Müşterinin gereksinimlerine göre ürünün özellikleri , ürünün ara yüzü , ürünün kabiliyetleri gibi konuların tasarımına başlanır . Bu aşamada herhangi bir kodlama söz konu değildir . Tasarım aşaması Yüksek düzeyde tasarım (Mimari tasarım) ve Detaylı tasarım olarak ikiye ayrılır . Mimari tasarım yazılımın parçalarının genel yapısı ve etkileşimleri ile ilgilenir . Sonucunda mimari tasarım dokümanları oluşturulur . Detayları tasarımda mimari tasarımdaki dokümanlar revize edilir . Tasarım ve analiz arasındaki fark “Ne?” ve “Nasıl?” sorularıdır . Analiz aşamasında sorunun ne olduğuna bakarken tasarımda bu sorunu nasıl çözeceğimizi düşünürüz . Tasarımda kullanılan bir çok teknik vardır , bunlardan en yaygın kullanılanı soyutlamadır (Abstraction) . Soyutlama problemi daha kolay bir hala getirmek için kullanılır (ek sorunları görmezden gelip ana probleme odaklanmak gibi) . Tasarımda temel araç olarak modelleme kullanır . Modelleme statik modelleme ve dinamik modelleme diye iki ayrı yapıdadır . Statik model , programın çalışması sırasında değişmeyen yönlerini göstermek için kullanılırken (sınıf modelleri gibi) , dinamik model programın çalışırken ki işleyişi göstermek için kullanılır (sıra diyagramları gibi) .

4.) Gerçekleştirim (kodlama ve test) : Yazılım yaşam döngüsünün dördüncü adımıdır . Bu aşamanın asıl amacı tasarım aşaması biten ürünün kodlanmasıdır . Kodlama bittikten sonra test aşaması vardır . Test aşaması da kodlama kadar önemlidir .

5.) Teslim ve bakım : Yazılım yaşam döngüsünün beşinci ve son adımıdır . Bu aşamanın asıl amacı bütün test aşamalarını bitirip yazılım son halini kullanıcıya sunulmasıdır . Bundan sonra yazılım tüm yaşamı boyunca çalışacaktır . Kullanıcılardan geri bildirimler alıp yazılımı ona göre şekillendirmeniz gerekir (Hata giderme , güncelleme gibi) .

Bu yazılım yaşam döngüsü modellerini temel aşamalarıdır . Yazılım yaşam döngüsünün en iyi sonucu vermesi ve daha aktif çalışması için Yazılım Belirtim Yöntemleri ve Yazılım Süreç Modelleri kullanılır .

Yazılım belirtim yöntemleri : Süreç akışı için kullanılan belirtim yöntemleri , Süreç tanımlama yöntemleri ve Veri tanımlama yöntemleri olmak üzere üçe ayrılır . Süreç akışı için kullanılan belirtim yöntemleri , süreçler arası ilişkilerin ve iletişimin gösterildiği yöntemlerdir (Veri Akış Şemaları , Yapısal Şemalar , Nesne/Sınıf Şemaları gibi) . Süreç tanımlama yöntemleri , süreçlerin iç işleyişini göstermek için kullanılan yöntemlerdir (Düz Metin , Karar Tabloları , Algoritma gibi) . Veri tanımlama yöntemleri , süreçler tarafından kullanılan verilerin tanımlanması için kullanılan yöntemlerdir (Nesne İlişki Modeli , Veri Tabanı Tabloları , Veri Sözlüğü gibi) .

Yazılım Süreç Modelleri (Processes) : Yazılım yaşam döngüsünde belirtilen adımların geliştirme aşamasında , hangi düzen ya da sırada , nasıl uygulanacağını tanımlayan modellerdir . Karmaşıklığı azaltıp krizleri önler . Bazı yazılım süreç modelleri şu şekildedir : Kodla ve Düzelt (Code and Fix) , Şelale Modeli (Waterfall Model) , V Modeli (V-shaped Model) , Evrimsel Geliştirme (Evolutionary Development) , Prototipleme (Prototyping) , Spiral Model , Formal Sistem Geliştirme (Formal System Development) , Yeniden kullanıma yönelik geliştirme (Re-use based development) , Artımlı Geliştirme (Incremental Development) , Birleşik Süreç (Unified Process) , Çevik Modeller (Agile models : XP , Scrum)

1. Kodla ve Düzelt (Code and Fix) : Bu modelde çoğunlukla resmi olmayan bir ürün fikriyle başlar ve program ürün hazır olana kadar ya da gerekli zaman bitene kadar kodlama yapılarak devam eder .
2. Avantajları :

* Herhangi bir planlamaya ihtiyaç duyulmaz .
* Çok küçük projelerde ya da kısa ömürlü prototiplerde uygulanabilir .
* Program aşamaları çabuk geçilir .
* Uzman görüşüne ihtiyaç düşüktür herkes bu modeli kullanabilir .

1. Dezavantajları :

* Kontrollü değildir .
* Kaynak planlaması yoktur .
* Bitiş süresi belli değildir .
* Hataların bulunması ve doğrulaması zordur .
* Kodları düzeltmek maliyetli olabilir .
* Kodlar kullanıcının ihtiyacını karşılamayabilir .
* Kodlar sonradan değiştirmek için planlanmadığından esnek değildir, değiştirilmesi zordur .

1. Şelale Modeli (Waterfall Model) : Bu modelde yazılım geliştirme süreci analiz , tasarım ,​​ kodlama , test , sürüm ve bakım​​ gibi adımlardan oluşur .​​ Geleneksel yazılım metotlarında bu adımlar sıralı olarak işler . Her adım ,​​ başlangıç noktasında bir önceki adımın ürettiklerini bulur .​​ Kendi bünyesindeki değişikler doğrultusunda teslim​​ aldıklarını bir sonraki adımın kullanabileceği şekilde​​ değiştirir .​​
2. Avantajları :

* Kullanımı ve anlaması basittir .
* Yönetimi kolaydır .
* Projenin safhaları ayrı olduğundan iş bölümü ve iş planı projenin en başında net bir şekilde bellidir . Bu durum projenin yönetimini de oldukça kolay hale getirir .
* Şelale Modeli çok küçük ve gereksinimleri çok iyi anlaşılmış projelerde iyi çalışır .

1. Dezavantajları :

* Karmaşık ve nesne yönelimli projeler için uygun değildir .
* Devam eden ve uzun projeler için zayıftır .
* Projede oluşabilecek her türlü değişime elverişsiz , katı bir modeldir . Yapılan her değişiklik maliyeti büyük oranda arttırır .
* Müşteri memnuniyetini sağlamak çok zordur çünkü gelişim ve değişime açık bir model değildir .
* Model safhalardan oluştuğu için ürün son safhada tamamlanır , gereksinimlerin iyi tanımlanmadığı müşterinin ne istediğinin anlaşılmadığı bir projede bu durum projenin bittikten sonra iptal edilmesine ve başka gerginliklere sebep olmaktadır .

1. V Modeli (V-shaped Model) : Bu modelde şelale (waterfall) modelinin gelişmiş hali olarak düşünülebilecek bir yazılım geliştirme süreci sunar . Doğrusal bir yönde ilerlemek yerine , süreç adımları kodlama evresinden sonra yukarıya doğru eğim alır ve tipik V şeklini oluşturur . V Model geliştirme yaşam çevriminin her bir evresi arasındaki ilişkileri gösterir . Yatay ve dikey açılar zaman veya projenin tamamlana bilirliğini ve soyut seviyeyi gösterir .
2. Avantajları :

* Verification ve validation planları erken aşamalarda vurgulanır .
* Verification ve validation sadece son üründe değil tüm teslim edilebilir ürünlerde uygulanır .
* Proje yönetimi tarafında takibi kolaydır .
* Kullanımı kolaydır .

1. Dezavantajları :

* Aynı zamanda gerçekleştirilebilecek olaylara kolay imkan tanımaz .
* Fazlar arasında tekrarlamaları kullanmaz .
* Risk çözümleme ile ilgili aktiviteleri içermez .
* Yazılım da diğer sistemler gibi zamanla evrimleşir .
* Geliştirme devam ettikçe iş ve ürün gereksinimleri de değişkenlik gösterebilir .
* Son ürüne ulaşma düz bir çizgi ile ifade edilemez .

4.) Evrimsel Geliştirme (Evolutionary Development) : İlk tam ölçekli modeldir . Coğrafik olarak geniş alana yayılmış , çok birimli organizasyonlar için önerilmektedir (banka uygulamaları) .​​ Her aşamada üretilen ürünler , üretildikleri alan​​ için tam işlevselliği içermektedirler .​​ Pilot uygulama kullan , test et , güncelle diğer birimlere taşı .​​ Modelin başarısı ilk evrimin başarısına bağımlıdır . Keşifçi geliştirme (exploratorydevelopment) ve Atılacak prototipleme (throw-awayprototyping) olmak üzere iki çeşit evrimsel gelişim vardır . Keşifçi geliştirmenin asıl amacı müşterinin gereksinimlerini incelemek için müşteri ile çalışıp son sistemi teslim etmektir . Atılacak prototiplemenin asıl amacı sistem gereksinimlerini anlamaktır .

1. Avantajları :

* Kullanıcıların kendi gereksinimlerini daha iyi anlamalarını sağlar .
* Sürekli değerlendirme erken aşamalardaki geliştirme risklerini azaltır .
* Hatalar azalır .

1. Dezavantajları :

* Sürecin görünürlüğü azdır (düzenli teslim edilebilir ürün yoktur) .
* Sistemler sıklıkla iyi yapılandırılmaz (sürekli değişiklik yazılımın yapısına zarar verir) .
* Bakımı zordur .
* Yazılım gereksinimini yenilemek gerekebilir .

5.) Prototipleme (Prototyping) : Gereksinimler hızlıca toplanarak işe başlanılır. Geliştiriciler ve kullanıcılar aynı masa etrafında buluşarak yazılımdan elde edilecek bütün çıktılara, bu çıktılar için gerekli girdilerin nasıl sağlanacağına, nasıl korunacağına, hangi işlemlere uğrayacağına karar verirler.​​ Daha sonra hızlıca yapılan bir tasarım ile yazılımın kullanıcıya yansıyacak yönünü aktaran bir​​ ilk örnek​​ üretilir. Prototip kullanıcının kullanımına ve değerlendirilmesine sunulur. Bu değerlendirmelere bakılarak​​ ilk örnek​​ üzerinde gerekli​​ değişiklikler yapılır. Prototipin yeni hali kullanıcı tarafından yeniden değerlendirilir. Böylece kullanıcının istediği yazılıma iyice yaklaşılmış bir​​ ilk örnek​​ üzerinde yazılımın​​ neler yapacağı konusunda kullanıcı ile anlaşmaya varılır.​​ Doğrusal modelin döngüsel versiyonudur.​​ Bu modelde, gereksinim analizi ve prototipleme için tasarım yapıldıktan​​ sonra, geliştirme​​ süreci başlatılır.​​ Prototipleme yaratıldıktan sonra, müşteriye değerlendirme için verilir.​​ Müşteri paketi test eder ve düşüncelerini, ürünü müşterinin tam beklentilerine göre düzenleyen geliştiriciye iletir.​​ Sınırlı sayıdaki yinelemelerden sonra, son yazılım paketi müşteriye verilir.​​ Bu metodolojide, yazılım müşteri ve geliştirici arasında periyodik bilgi gidip gelmeleri sonucunda gelişir .

1. Avantajları :

* Kullanıcı sistem gereksinimlerini görebilir .
* Karmaşa ve yanlış anlaşılmaları engeller .
* Yeni ve beklenmeyen gereksinimler netleştirilebilir .
* Risk kontrolü sağlanır .

1. Dezavantajları :

* Belgelendirmesi olmayan hızlı ve kirli (quick and dirty) prototipler .
* Prototip hedefleri net değilse kod hackleme ya da jenga başlar .
* Düzeltme aşaması atlanırsa, düşük performansa yol açar .
* Müşteri prototipten de son ürün gibi görünüm ve etki bekler .

6.) Spiral Model : Spiral modeli aynı safhalara geri dönülmesinin bir zorunluluk olduğunu vurgular , Spiral modeli şelale modelinde yok sayılan riskleri göz önünde bulundurur . Proje çevrimlere ayrılır ve her bir çevrimin riskleri ayrı ayrı ele alınır . Çağdaş modellere son derece yakındır .

Planlama: Her aşamada olan ara ürün için bir planlama yapılır.

Risk Analizi: Risklerin araştırılması, belirlenmesi ve çözülmesi.

Üretim: Ara ürünün/ürünün üretilmesi.

Kullanıcı Değerlendirme: Oluşturulan ara ürünün sonucunda kullanıcıdan alınan geri dönütlerin değerlendirilerek diğer aşamaya geçilmesi.

1. Avantajları :

* Kullanıcılar sistemi erken görebilirler .
* Geliştirmeyi küçük parçalara böler . En riskli kısımlar önce gerçekleştirilir .
* Pek çok yazılım modelini içinde bulundurur .
* Riske duyarlı yaklaşımı potansiyel zorlukları engeller .
* Seçeneklere erken dikkate odaklanır .
* Hataları erken gidermeye odaklanır .
* Yazılım-donanım sistemi geliştirme için bir çerçeve sağlar .

1. Dezavantajları :

* Küçük ve düşük riskli projeler için pahalı bir yöntemdir .
* Komplekstir (karmaşık) .
* Spiral sonsuza gidebilir .
* Ara adımların fazlalığı nedeniyle çok fazla belgelendirme gerektirir .
* Büyük ölçekte projeler .
* Kontrat tabanlı yazılıma uymaz .
* Yazılımın içten geliştirileceğini varsayar .
* Kontrat tabanlı yazılımlar adım adım anlaşma esnekliğini sağlamaz .
* Öznel risk değerlendirme deneyimine dayanır .
* Yüksek riskli öğelere yoğunlaşmak, yüksek riskli öğelerin doğru belirlenmesini gerektirir .

7.) Formal Sistem Geliştirme (Formal System Development) : Formal Sistem Geliştirme Modeli​​ yazılım tasarım ve gerçekleştirmesiyle ilgili matematiksel bir tekniktir . Bu modelin temelinde karmaşık sistemleri geliştirme ve program geliştirmeye destek yatar . Formal Sistem Geliştirme Metodu​​ kullanıcı sistemi kullanmaya başladığında karşısına çıkan belirtim hatalarını minimize eder . Formal belirtim , tasarım ve geçerleme kullanarak yazılımda doğruluğun geliştirilmesini vurgular .​​ Yazılım artımlarla geliştirilir .​​ Sürekli tümleştirme vardır ve fonksiyonellik tümleştirilen yazılım artımları ile artar .​​ Felsefesi pahalı hata ayıklama işlemini engellemek için kodu ilk yazarken doğru yazmak ve test aşamasından doğruluğunu sağlamaktır .​​

1. Avantajları :

* Yazılımdaki belirsizlikleri , eksiklikleri ve uyumsuzlukları saptar .
* Hatasız yazılım geliştirme imkanları sunar .
* Her iterasyondan sonra aşamalı olarak artan efektif çözümler sunar .
* Karmaşık değildir .

1. Dezavantajları :

* Çok zaman alan ve pahalı bir yöntemdir .
* Kullanımı esnasında teknik olmayan personelle iletişim mekanizması zor işler .
* Sadece birkaç geliştirici bu modelin uygulamasıyla ilgili temel bilgilere sahip olması için yaygın eğitim gerektirir .

8.) Yeniden kullanıma yönelik geliştirme (Re-use based development) : Organizasyon tarafından daha önce hazırlanmış veya dışarıdan temin edilmiş yazılımların (veya yazılım parçalarının) kullanılması ile geliştirme yapılması son yıllarda popülaritesi artan bir yaklaşımdır . Organizasyonların olgunlukları arttıkça , bu tür uygulamalar yapabilmek için alt yapı kurulmuş olmaktadır .

1. Avantajları :

* Kaynak kontrolü mümkündür .
* Maliyet denetimi yapmak mümkündür .
* Basit ve anlaşılırdır .
* Önceden oluşturulmuş sınıflar tekrardan kullanılabilir .
* Kısa sürede yazılım geliştirilebilir .

1. Dezavantajları :

* Gereksinimleri anlamak güçtür .
* Pahalıdır .
* Uzmanlık gerektirir .
* Başarım garantisi yoktur .

9.) Artımlı Geliştirme (Incremental Development) : Eğer bir müşterinin ürünlerinde değişikliğe ihtiyaçları varsa , artımlı model ihtiyaç olan bu değişikliğe ayak uydurur .​​ Artırımsal model bir takvime bağlı olarak yazılımı parça parça geliştirip teslim etmeye dayanır. Her bir yeni kesim öncekinin üstüne bazı ek işlevlerin eklenmesini öngörür . Artırımsal model yazılım geliştirmenin kısıtlı sayıda çalışanla işin yapılmasını sağlama gibi bir üstünlüğü vardır . Ayrıca çalışanlar da her artırım geçildiğinde uygulama alanına ilişkin daha çok deneyim kazanmış olurlar . Bu modelde bir taraftan üretim bir taraftan da kullanım yapılır . Önceki modellerde ürünlerdeki değişiklikler göz önünde bulundurulmaz . Bu model doğal olarak yinelemelidir .​​ Yeniden kullanılabilir bir ürün ,​​ fonksiyonellik sağlamış bir şekilde tüm döngülerin sonunda ortaya çıkar .

1. Avantajları :

* Sistem için gerekli olan gereksinimler müşterilerle belirlenir .
* Gereksinimlerin önemine göre teslim edilecek artımlar belirlenir .
* Öncelikle en önemli gereksinimleri karşılayan çekirdek bir sitem geliştirilir .
* Erken artımlar prototip gibi davranarak , gereksinimlerin daha iyi anlaşılmasını sağlar .
* Tüm projenin başarısız olma riskini azaltır .
* En önemli sistem özellikleri daha fazla sınanma (test edilme) imkanı bulmuş olur .
* Divide and Conquer (Böl ve Yönet) yaklaşımıdır .

1. Dezavantajları :

* Artımları tanımlamak için tüm sistemin tanımlanmasına ihtiyaç vardır .
* Gereksinimleri doğru boyuttaki artımlara atamak bazen zor olabilir .
* Deneyimli personel gerektirir .
* Artımların kendi içlerinde tekrarlamalara izin vermez .

10.) Birleşik Süreç (Unified Process) : Nesneye dayalı yazılım geliştirmek için var olan yöntemlerin deneyimler sonucu kabul gören en iyi özellikleri bir araya getirilmesiyle türetilmiş yazılım geliştirme süreci (The Unified Process - UP) oluşturulmuştur . Yinelemeli , arttırmalı ve evrimsel aynı zamanda da risk güdümlüdür .

1. Avantajları :

* Değişen isteklere uygunluk .
* Erken geri besleme .
* Büyük sistemlerde çözme kolaylığı .
* Her iterasyonda deneyim kazanılması .
* Risklerin erken giderilmesi .
* Erken ürün elde etme ve takımda moral yükselmesi .

1. Dezavantajları :

* Risk yönetimi zayıftır .
* Belgelendirme yükü ağır olduğundan pahalı olabilir .
* Karmaşıktır .

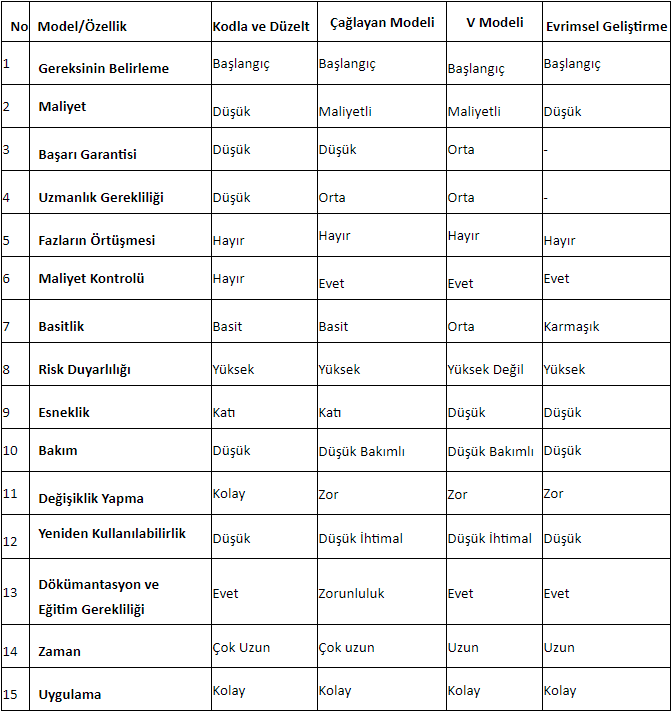
11.) Çevik Modeller (Agile models : XP , Scrum) : İnteraktif geliştirme temelli bir grup yazılım geliştirme metodolojisine dayanır . Burada gereksinimler ve çözümler kendinden örgütlü olan farklı grupların ortak çalışmasıyla olgunlaşır . Agile metotları genellikle sık denetim ve adaptasyonun teşvik edildiği disipline edilmiş bir proje yönetim sürecini , takım çalışmasının , kendinden örgütlenmenin ve izlenebilirliğin özendirildiği liderlik felsefesini, kaliteli yazılımların hızlı biçimde geliştirilmesinin hedeflendiği en iyi mühendislik uygulamalarını ve yazılım geliştirme ile müşteri ihtiyaçlarını ve firma amaçlarını yan yana getiren business yaklaşımı destekler .​​ Agile metotları bir yazılım geliştirme yaklaşımı değil , geliştirme süreçleri topluluğudur .​​ Çevik yazılım bir yandan bir değer sistemini , diğer yandan da somut yazılım metotlarını içerir . Çevik yazılıma , yazılım sektöründe yeni bir filozofi akımı , ya da yeni bir yazılım meta modeli olarak bakabiliriz . Bu yeni filozofikin somut örnekleri arasında Extreme Programming , Scrum ve Lean Development bulunmaktadır . Süreç ve araçlar arası etkileşim , kapsamlı belgelendirme üzerinden yazılım geliştirme , uzlaşılan taahhüt üzerinden müşteriyle birlikte çalışma , değişiklik yönetimini plan üzerinden yönetmek gibi değerleri vardır . Teslimatı hızlı ve aralık yaparak müşteri memnuniyeti sağlamak , çalışan yazılımın aylar yerine haftalar süren zamanda teslim etmek , sürecin esas değerlendirme kriteri çalışan sistemdir , gereksinimlere sonradan eklenen değişiklikler mutlaka dikkate alınır , business ve teknik ekip arasında sıkı ve günlük çalışma , en iyi iletişim yolu yüz yüze görüşme kabul edilir , proje güvenilir ve motivasyonu yüksek kişiler üzerine kurulur , teknik başarı ve iyi tasarıma düzenli iltifat , basitlik , kendinden örgütlü takımlar , değişen koşullara adaptasyon sağlama gibi prensiplere sahiptir .

1. Avantajları :

* İnsanın doğal eğilimine çok yatkındır öğrenim gerektirmez adaptasyon hızlıdır .
* Kısa döngüler dolayısı ile takım elemanlarında motivasyon çok yüksektir . Verim artışı yaşanır .
* Sık çıktı üretip geri besleme aldığından kaynağı müşteri ihtiyaçlarına ve sonuca analiz etmeye odaklanır.
* Plan aşamasında ayrıntılı plan yerine iterasyonun planı yapılır .
* Değişime açıklık ve esneklik en üst düzeydedir .
* Sürdürülebilir Kalite .
* Proje planlama ve yürütme bir arada .
* Takım oyunu .

1. Dezavantajları :

* Kurumsal bir yapıda uygulaması gerçekten zor .
* Belgelendirme hakkındaki taşları yerinden oynatan yaklaşımı .
* Sürekli değişen ihtiyaçlar dolayısı ile aşırı çalışma .
* Ürünün başarısı = projenin başarısı dolayısı ile kariyer riski .
* Takım üzerindeki hedef baskısı .

KARŞILAŞTIRMA





KAYNAKÇA :

* <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>
* <https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>
* <https://medium.com/@brfn.kcr26/yazilim-geli%CC%87%C5%9Fti%CC%87rme-ve-s%C3%BCre%C3%A7-modelleri%CC%87-2131ea5f09b2>
* <https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742>
* <https://caglartelef.com/yazilim-yasam-dongusu/>

Adı Soyadı : Utku Baran Keşanlı

Numara : 200601041

Linkedin = <https://www.linkedin.com/in/utku-baran-ke%C5%9Fanl%C4%B1-0959b2209/>

Medium = <https://medium.com/@utkubarankesanli>

GitHub = <https://github.com/UtkuKesanli>

Linkedin (makale) =

Medium (makale) =

GitHub (makale) =