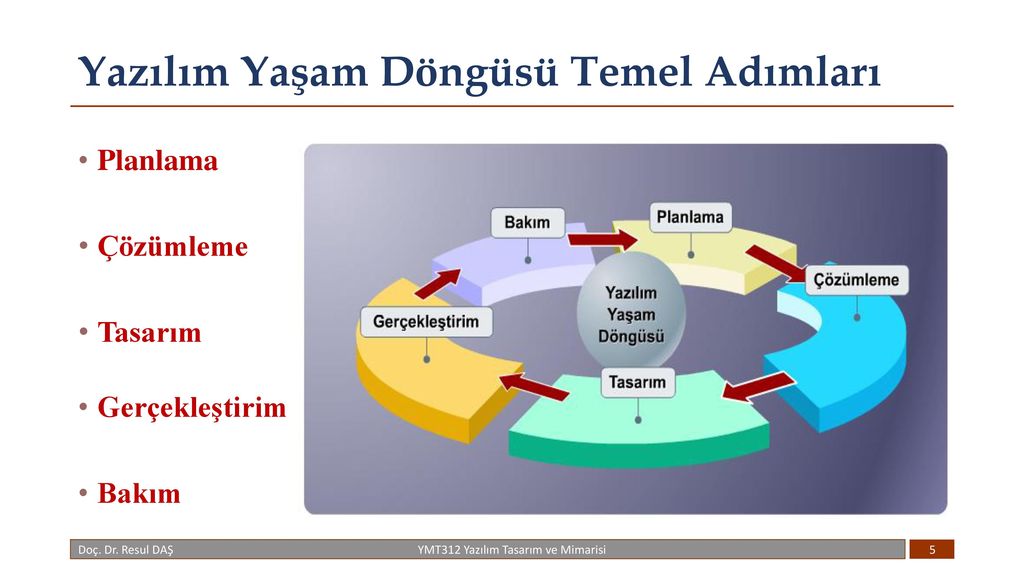
YAZILIM YAŞAM DÖNGÜSÜ

Yazılım yaşam döngüsünden bahsetmeden önce yazılım ne demek bir bakalım. Yazılım, elektronik cihazlara atanmış olan bir görevi yerine getirebilmesi için bilgisayar dilinde (Makine Dili) geliştirilen komutlara denir. Genel olarak yazılımlar, geliştirilirken belirli birtakım süreçlerden geçerler. İngilizcesi Software Development Life Cycle (SDLC) olarak adlandırılan bu süreç yazılım projelerinin daha iyi sonuçlandırılmasını sağlar. Geliştirmekte olduğumuz yazılım projesi sadece kodlamadan ibaret değildir. Proje geliştirirken planlama, analiz-çözümleme, tasarım, gerçekleştirim ve bakım aşamaları yer almaktadır. Bu aşamalar bir kere gerçekleştikten sonra proje tamamlanmış olmuyor, projede herhangi bir hata oluştuğun da bir önceki adımların gözden geçirilmesi gerekiliyor işte bu yüzden bu aşamaların bir döngü halinde düşünülmesi gerekilir. Bu döngü proje tamamlandıktan sonra da müşteriden gelecek yeni gereksinimler, hata düzeltmeleri ve projeye eklenecek yeni modüller için devam etmektedir. Bu döngüye yazılım geliştirme yaşam döngüsü adı verilmektedir.



Planlama : Gereksinim analizi yapılmasıyla başlanılır. Gereksinim analizi müşterinin istekleri doğrultusunda ortaya çıkacak olan yazılım hakkında ilk bilgilerin alındığı adımdır. Bu adımda müşteri iyi bir şekilde dinlenmeli ve istekler tam ve kesin olarak belirlenmelidir. Anlam kargaşası yaşamamak için müşteriye sorular yöneltilmelidir. İkinci adımda ise fizibilite çalışması yapılır. Fizibilite, yatırım kararlarının öncesinde durum değerlemesidir. Fizibilite çalışmaları ile topladığımız veriler rapor haline getirilir ve bu rapor (Fizibilite Rapor) ; bir iş veya yatırıma başlama aşamasında ekonomik, teknik ve finansal araştırmaların yapılmasını ve bu araştırmaların sonucunda yatırımın karlı olup olmadığını ortaya koyar. Raporun değerlenmesi sonucunda karlı bulunan yatırımlar için fizibildir denilir. Fizibil denmesiyle birlikte Yazılım Yaşam Döngüsü Temel Adımların da ikinci kısım olan Analiz-Çözümleme ye geçilir.

Analiz- Çözümleme : Yazılım işlemleri ile sistem gereksinimlerinin ayrıntılı olarak çıkarıldığı aşamadır. Bur da kastettiğimiz şey geliştireceğimiz yazılımın bilgisayarların sahip olduğu donanımda çalışıp çalışmayacağını gösteren yani yazılımın çalışması için gereken donanım bilgilerinin ayrıntılı olarak sunulduğu aşamadır. Bu aşama temel UML diyagramlarının çizilmeye başlanıldığı ilk adımdır.( Use Case, Activity, Class …)

Tasarım : Tasarım aşaması önemli bir süreçtir. Bu sürecin sonunda çıkan tasarım bir sonraki aşamalar olan kodlamanın ve testin temelini oluşturduğu için dikkat edilmeli, fazladan zaman ayırılmalıdır. Çözümleme aşamasından sonra belirlenen gereksinimlere çözüm olacak yazılım ilk ve temel halinin oluşturulması aşamasıdır.

Müşteri gereksinimlerinde sürekli değişim olması muhtemeldir, bu değişimlere değişen hedef problemi denir. Bu değişim yazılım ürününü kötü etkileyebilir. Bu sebeple tasarımın birinci amacı basitlik, anlaşıla bilirlik olmasıdır. Çünkü anlaşılır ve basit bir tasarım hem kodlamada hem de sonraki değişikler de kolaylık sağlar. Bu aşama 2’ye ayrılır. Bunlar:

Mantıksal Tasarım: Bir dizgenin bölümleri arasındaki çalışma ilişkilerini, donanımla ilgili gerçekleştirmeye birincil önem vermeksizin, simgesel mantık yaklaşımıyla tasarlamayı amaçlayan çalışma alanı.

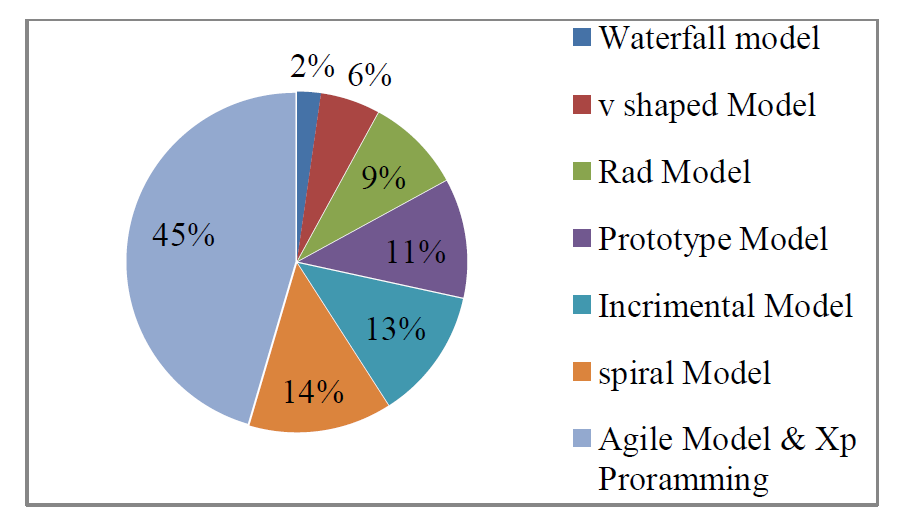
Fiziksel Tasarım : Yazılımı içeren bileşenler ve bunların ayrıntılarını içerir.

Gerçekleştirim : Bu aşama kodlamanın gerçekleşeceği aşamadır. Kodlama işlemi planlama veya çözümleme aşamasında belirtilen programlama dili, geliştirme ortamı ve teknolojilerin yardımıyla yapılır. Kodlama işlemi yaparken temiz kod yazmaya özen göstereceğiz, eğer yazdığımız kodlar kod karmaşıklığına sebep olursa ve artmaya devam ederse takımların verimliliği düşer ve sıfıra yaklaşır. Bu sefer takımda ki herkes verimliliği arttırmaya çalışır ama büyük bir baskı altında kalırlar. Öyle ki verimliliği sıfıra daha da yaklaştıracak şekilde kod karmaşası yaratmaya devam ederler. Bu tarz sorunlar yaşanmaması için temiz kod (clean code) yazımında belli başlı kurallara uyulması gerekilir. Gerçekleştirim aşaması süresince ve sonrasında test yapılır ama test sadece gerçekleştirim ve sonrasında yapılmaz test her aşamanın sonunda yapılır, her aşama kendine ait test kısmı içerebilir. Her aşamada test yapmamız hata yapma oranımızı düşürecektir.

Bakım : Bakım aşaması yazılım yaşamı boyunca sürer. Bakım deyince aklımıza gelmesi gereken ögeler “Hata giderme, yeni eklenti yapmak” olacaktır.

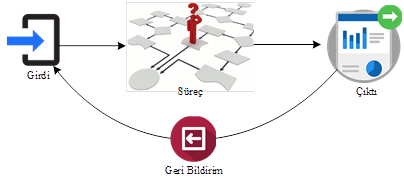
YAZILIM YAŞAM DÖNGÜ MODELLERİ

Yazılım geliştirmenin zorlukları vardır bu zorlukların beraberinde gelen problemler şirketler için sorun oluşturur çünkü kalitesiz ürün geliştiren şirketler itibar, müşteri ve gelir kaybı riski ile karşılaşabiliyor ve hatta bu durum şirketlerin geleceğini de riske atıyor. Bu sorunların yaşanmaması için planlı, düzenli, disiplinli çalışmak ve kaliteli ürünler ortaya koyabilmek için yazılım süreçlerini takip edebilecek modeller geliştirmeye başlanıldı. Bu modellere yazılım yaşam döngü modelleri adı verilir. Bu modellerin asıl temel hedefi; proje başarısı için, yazılım yaşam döngüsü boyunca izlenmesi önerilen mühendislik süreçlerini tanımlamaktır.



Yazılım Yaşam Döngü Modellerinin kullanım oranları

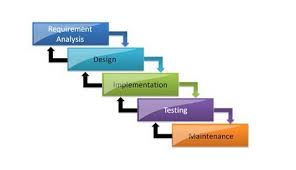
Gelişigüzel Model : Geliştirme ortamında herhangi bir model ya da yöntem kullanılmaz. Çoğunlukla tek bir yazılımcının ürettiği ve kendine bağlı yazılım geliştirme şeklinde yapılır. İzlenebilir tekrarlanabilir başarılar elde edebilmek için bir süreç modeline ihtiyacımız var. Başta dediğim gibi bir model ya da yöntem kullanılmadığı için yaptığı yöntemin izlenebilirliği ve tekrarlılığı yoktur. Geliştiren yazılımcı bile aradan belirli zaman geçtikten sonra anlayamayacağı ve müşteriden gelecek yeni gereksinimlerle projeyi değiştirme zorluğu yaşayacağı ortamlarda ki üretim tarzıdır. Yani basit programlama yöntemidir.



Barok Modeli : Yaşam döngüsü temel adımlarının doğrusal olarak gerçekleşmesini öngörür. Yazılım yaşam döngü temel adımlar arasında döngü yoktur. Belgelemeyi (Dokümantasyon) ayrı bir süreç olarak ele alır. "Dokümantasyon nedir?", dokümantasyonu anlamak için önce "doküman nedir?" sorusuna cevap aramalıyız. Doküman latince kökenli bir kelimedir belge anlamına gelmektedir. Dokümantasyon ise fransızca kökenli bir kelimedir karşılığı belgelemedir ; bir çalışma için gerekli belgeleri arama ve sağlama, belgelere dayandırma anlamına gelmektedir. Dokümantasyon yazılımın geliştirilmesi ve testinden hemen sonra yapılması öngörülür. Kodlamanın yapıldığı aşamaya (Gereksinim) daha fazla önem veren bir modeldir.



Çağlayan Yaşam Döngü Modeli : Günümüzde geleneksel model olarak anılmakta ve kullanımı azalmaktadır. Özellikle müşteri gereksinimleri iyi dinlenilmiş, tam ve kesin olarak belirlenmiş, üretimi az zaman gerektiren projeler için uygun bir modeldir. Karmaşık ve nesne yönelimli projeler için uygun değildir. Gereksinim tanımları çoğu kez net olarak yapılmaz bu tür eksiklikler ilerleyen aşamalarda ortaya çıkar. Bu gibi eksikliklerin düzeltilip, giderilmesi ciddi maliyetlere sebep olabilir mesela analiz aşamasında ki bir hatanın düzeltilmesi çok daha düşük maliyete sebep olurken bu problem testte yakalanırsa belki de tekrar bütün süreçlerin ele alınmasını gerektiriyor ve bu maliyet artışını beraberinde getiriyor. Yazılım geliştirme maliyetini yükselten diğer bir durum ise genelde müşterinin, yazılım geliştirme sürecinde yer almamasından kaynaklanır bunun sebebi yazılım ürünü tamamlandıktan sonra geri dönüşlerin olma olasılığının fazla olması. Çağlayan yaşam döngü modeli barok modelin aksine belgeleme işlevini ayrı bir aşama olarak ele almaz ve üretimin doğal bir parçası olarak görür ve aşamalar arasında döngü vardır. Her aşama bir önceki aşama bitmeden başlamaz. Yazılımcılar çoğunlukla projenin kod kısmıyla uğraşmak isterler ama bu modelde analiz ve tasarım aşamalarının daha ayrıntılı incelenmesi gerektiğinden yazılım sürecinin çoğu bu aşamalarda geçer, kodlama kısmı ise hızlı geçer bu yüzden de ekipte mutsuzluk gözükür ve çalışma motivasyonları düşer aynı şekilde şirkette bu durumdan etkilenir o yüzden çağlayan yaşam döngü modelinin kullanımı giderek azalmaktadır.

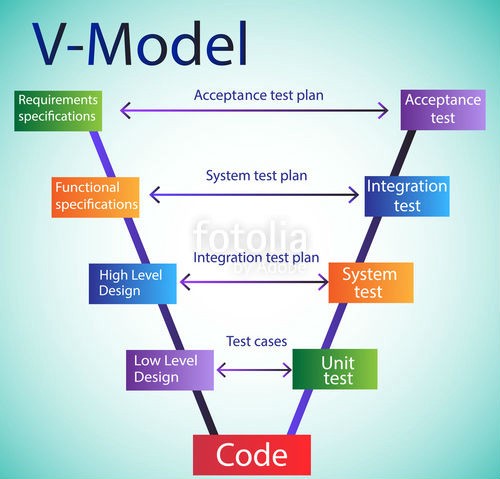


V Süreç Modeli : V süreç modelinin sol tarafı üretim, sağ tarafı ise sınama işlevleri ile ilgilidir. Test işlevlerinin herhangi birinde hata çıkarsa yatay doğrultusunda ki üretim kısmına gidilerek bu sorun çözülür. Eş zamanlı gerçekleştirilebilecek olaylara kolay imkan tanımaz. V süreç modeli, çağlayan yaşam döngü modelinin gelişmiş hali olarak düşünülebilecek bir yazılım geliştirme süreci sunar. Doğrusal bir yönde ilerlemek yerine , süreç aşamalarının kodlama kısmından sonra yukarı doğru eğim alır ve üretim işlevlerinde ki her aşamanın kendine ait test kısmı karşısında bulunur. V süreç modelinin temel çıktıları: Kullanıcı modeli, mimari model ve gerçekleştirim model olarak adlandırılan üç alt modeldir.

Kullanıcı Modeli : Geliştirme sürecinin kullanıcı ile olan ilişkileri tanımlanmakta ve sistemin nasıl kabul edileceğine ilişkin sınama belirtimleri ve planları ortaya çıkarılmaktadır.

Mimari Model : Sistem tasarımı ve oluşacak alt sistem ile tüm sistemin sınama işlemlerine ilişkin işlevler.

Gerçekleştirim Model : Yazılım modüllerinin kodlanması ve sınanmasına ilişkin fonksiyonlar.



Helezonik (Spiral) Model : Spiral geliştirme modeli dört ana bölüm içerir. Bunlar planlama, risk yönetimi, üretim ve kullanıcı değerlendirmesidir.

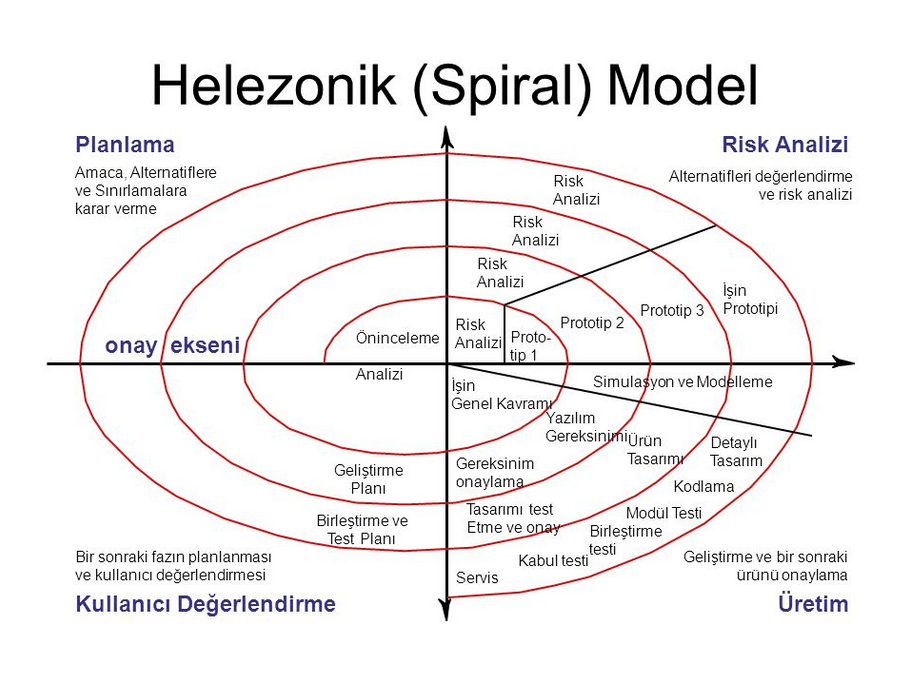
Planlama: Üretilecek ara ürün için planlama, amaç belirleme, bir önceki adımda üretilen ara ürün ile bütünleştirme.

Risk Çözümleme : Risk seçeneklerinin araştırılması ve risklerin belirlenmesi.

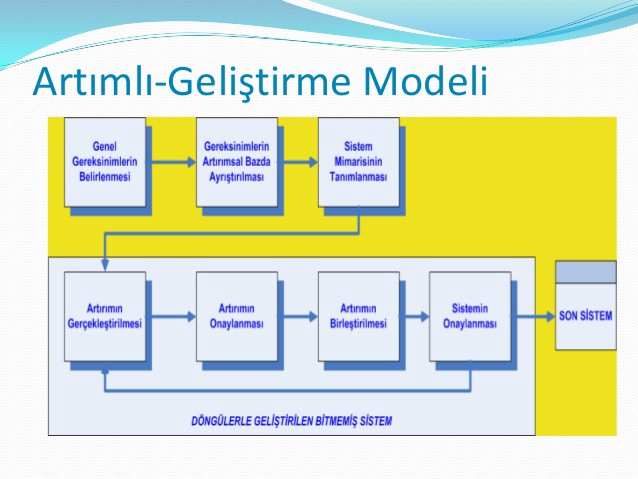
Üretim : Ara ürünün üretilmesi.

Kullanıcı Değerlendirmesi : Ara ürün ile ilgili olarak kullanıcı tarafından yapılan sınama ve değerlendirmeler.

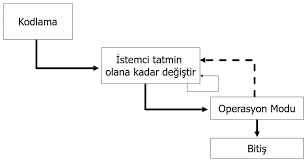
Her bir tam turunda prototipler (Prototype) ortaya çıkıyor ve her döngüsü ise bir fazı ifade eder. "Prototip nedir?" sorusunu şöyle cevaplayalım: Prototip, bir kavram ya da süreci test etmek ya da çoğaltılmış ve öğrenilecek bir şey gibi davranmaya yönelik inşa edilmiş, bir ürünün erken örneği, modeli ya da sürümüne denir. Prototipin amacı icadın kusurlarını, eksikliklerini ya da gerçekten pratik hayatta işe yarayıp yaramadığını anlamanızı sağlamasıdır. Bir final ürünü değil ama ürünün o aşamada finale mümkün olan en yakın halidir. Oluşan prototipler müşteriye sunulur ve bu prototipler sayesinde müşteri nelerin eksik veya tam olduğunu belirtir, sonraki fazda ise bu eksiklikler giderilerek yeni prototipler oluşturulup tekrardan müşteriye sunulur, asıl ürüne gelene kadar bu döngü devam eder. Uygulamaya geçmeden önce ki son prototipe ise operasyonel prototip (operational prototype) denir. Bu prototipler projenin büyüklüğüne göre artış veya azalış gösterebilir. Başta dediğim gibi her döngü bir fazı ifade eder faz sayısı ne kadar artarsa y ekseninde o kadar genişleme olacak buda maliyetin ona göre artacağını gösteriyor aynı şekilde projeye verilen zaman da artmış oluyor. Helezonik model de risk çözümleme aşaması ön planda tutuluyor bu sebeple küçük ve düşük riskli projeler için çok pahalı bir modeldir.

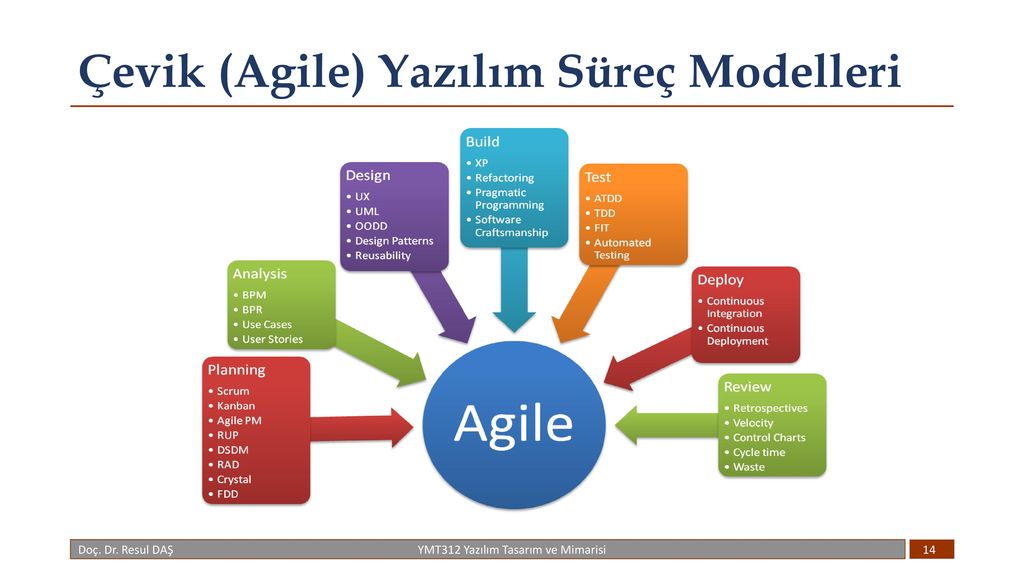


Artımsal Geliştirme Süreç Modeli : Uzun süren projeler için eksiklikler çıktıkça üzerine ekleme yapabileceğimiz projeler için uygundur. Kullanım ve üretim eş zamanlı gerçekleşir. Artımlı geliştirme modelinde proje parçalara bölünür ve kullanıcının önceliğine göre bu parçalar sıralanır. Deneyimli personel gerektirir. Modelde bir döngü söz konusudur. Planlama aşaması ile başlar ve temel adımların bulunduğu döngüye giriş yapar. Uygulama aşamasına gelindiği vakit sistem ikiye ayrılır. Bir tarafta uygulama müşteriye gösterilirken diğer bir tarafta uygulamanın testleri yapılmaya devam eder. Testler sonucunda çıkan ve müşteriden gelen eksikliklerle proje tekrar döngü içine girip eksikliklerin giderilmesi sağlanılır. Tabi bunlar gerçekleşirken planlamada da yeni gereksinimler olabilir. İşte bütün bunlar artırımlı modelde her seferinde arttırıla arttırıla bir iterasyon şeklinde devam eder ve her yenilik bir sonraki iterasyonda sürece dâhil edilir. Artımsal modelin çağlayan yaşam döngü modelinden ayıran en büyük özelliği hata veya yeni gereksinimler çıkması sonucu maliyetinde büyük bir değişim olmamasıdır.

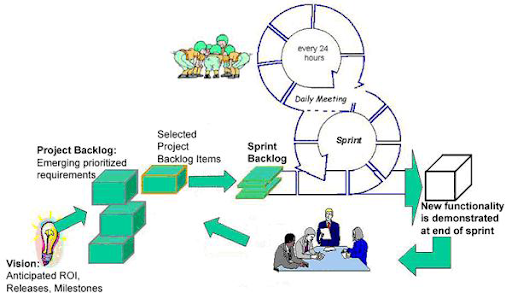


Kodla ve Düzelt Yaşam-Döngü Modeli : Büyük projeler için kullanılmaz küçük projelerde ya da kısa ömürlü prototipler için kullanılabilir. Direkt ürün gerçekleştirilir. Herhangi bir yöntem içermeyen, dokümantasyon gerektirmeyen proje türüdür. Teknik kullanılmadığı için spagetti kod örnekleriyle karşılaşılabilir.



Çevik Yazılım Geliştirme : İletişim çok önemlidir iletişimi sağlıklı ve anlaşılır kılmak için yüz yüze görüşme tercih edilir. Sürekli müşteri ile iletişim halinde olunur herhangi bir bilgiye ihtiyaç duyulduğu vakit kolayca müşteri ile iletişime geçilip kolayca o bilgiye erişebilmeye olanak sağlar. Basitlik önemlidir, karmaşık yapılar onlar için aykırıdır. Basitlik kavramını benimsemekle yeni gereksinimlerde veya hatalar da yazılım ürününün kolay bir şekilde değişimini sağlayabilirler. Proje ekibi için kendi kendini organize eden takım yapısı gereklidir ve proje ekibinin sürekli birbirleriyle iletişim halinde olması öngörülür. Temel prensipler arasında müşteriyi memnun etmek vardır bunu da hızlı, devamlı ve kullanışlı yazılım üreterek sağlamayı amaçlarlar. Çevik yazılım modeli, kısa süreli ve küçük parçalar halinde yazılım geliştirilmesini öngörür. 

Scrum : Rugby sporundaki bir hücum taktiğinden adını almaktadır. Çevik yazılım geliştirme metodolojilerinden biridir. Scrum büyük projelerin işlevsel hale gelebilecek şekilde küçük parçalara ayrılıp her bir parçanın 2-4 hafta aralığında günlük 15-30 dakika süren ve ayakta gerçekleşen toplantılar ile birlikte her küçük parçayı tamamlanıp müşteriye sunulur. Gereksinimlerin tam ve net olarak anlaşıldığı karmaşık yapılar için en uygun metodolojidir.



REFERANSLAR

* <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/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742>
* <https://medium.com/@HayriRizaCimen/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-ve-s%C3%BCre%C3%A7-modelleri-70fdfb2f8f77>
* <https://caglartelef.com/yazilim-yasam-dongusu/>
* <https://www.elektrikport.com/teknik-kutuphane/yazilim-gelistirme-yasam-dongusu/11471#ad-image-0>
* <https://www.youtube.com/watch?v=u2rU8Wss4bw>
* <https://www.youtube.com/watch?v=SV6CfXWCKSo&t=2227s>