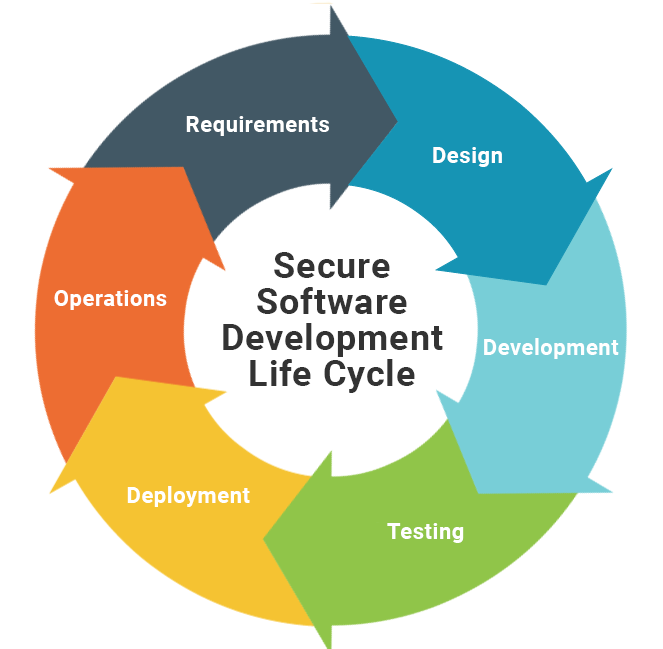
**Yazılım nedir?**

Sözlük anlamı “Bir bilgisayarda donanıma hayat veren ve bilgi işlemde kullanılan programlar, yordamlar, programlama dilleri ve belgelemelerin tümü.” olarak tanımlanmıştır. Gündelik hayatımızda kullandığımız bilgisayarlar, telefonlar, televizyonlar, tabletler, internet, akıllı ev teknolojileri ve sanayi vb. gibi pek çok alanda teknolojik her bir aletin bizim görmediğimiz tarafında bütün işlemleri hallederek hayatımızı kolaylaştıran donanıma can veren derlenmiş komutlar bütünü olarak tanımlayabileceğimiz bir üründür.

**Yazılım neden önemlidir?**

Daha önce bahsettiğim üzere hayatımızın her alanında bizimle olan yazılımın yokluğu durumda otomatik olarak gerçekleşen işler aksayacak, gündelik hayat sekteye uğrayacak, iş sektörleri büyük zarara uğrayacaktır. Fiziksel güç ihtiyacı artarak gereğinden fazla iş gücüne ihtiyaç duyulacaktır. En basitinden sabah sizi uyandıran alarmınız bile çalmayacak ve gün sizin için doğmayacaktır. Yazılım hayatımızın tam merkezinde yer alan ve bütün her şeye yön veren bir üründür. Bu ürünün dinamik olması çok önemlidir çünkü günümüzde değişmeyen tek şey değişimdir.



**Yazılım yaşam döngüsü (Software Development Life Cycle) nedir?**

Yazılımı donanıma can veren derlenmiş komutlar bütünü olan dinamik bir ürün olarak tanımlamıştık. Dinamik bir ürünün bir yaşam döngüsüne sahip olması gerekir. Yazılım yaşam döngüsü yazlımın geliştirme ve bakım sürecini kapsayan sistematik olarak belirli birkaç aşamadan oluşur. Bu aşamalar aşağıdaki gibidir:

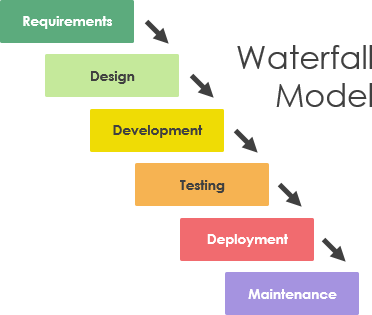
* **Planlama ve Gereksinim:** Müşteri gereksinimlerinin elde edildiği yazılım geliştirmenin ilk aşamasıdır. Bu aşamada gereksinimler belirlenir, fizibilite çalışmaları yapılır ve bunlara uygun planlama yapılır.
* **Analiz:** Sistemin gereksinim ve işlevleri detaylıca incelenerek kesin bir yargıya ulaşılır. Bu yargının belirli bir formatta sunulması gerekir. Her şeyin netleştiği aşamadır.
* **Tasarım:** Gereksinimlerin netleştiği aşamadan sonraki bu aşamada neyi nasıl yapacağımızı nereye varacağımızı belirleriz. Gereksinimleri karşılamaya yönelik yazılımın özelliklerini, yeteneklerini, arayüzlerini, temellerini oluştururuz. Bu aşama ikiye ayrılır:

1. **Üst seviye ve mimari tasarım:** Sistemin yapısı, modüllerin genel yapıları-organizasyon etkileşimleri, akış şemaları…
2. **Detaylı tasarım:** Doküman revizesi, yazılım bileşenleri, veri yapıları, ekran tasarımları…

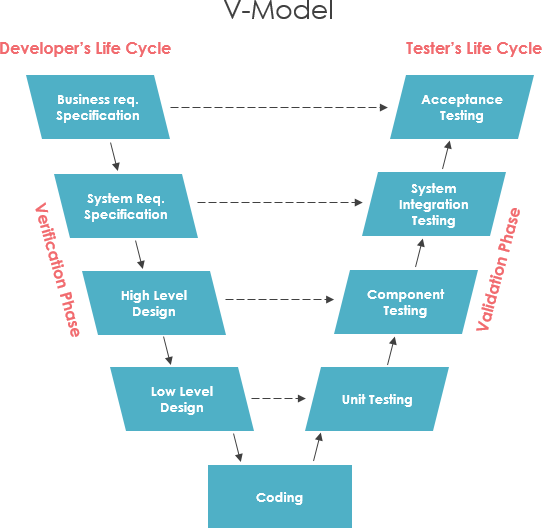
* **Gerçekleştirme:** Modüllerin kodlanarak birleştirildiği, test edildiği aşamadır. Kurulum çalışmaları yapılır. Analizi, planlaması ve tasarımı bitmiş olan yazılım ürününün artık kodlanmaya başlandığı aşamadır. Bu aşamanın sonunda test işlemleri iş, para ve zaman kaybetmemek açısından oldukça önem taşımaktadır. Herkes tarafından kolaylıkla okunabilen ve bakımı yapılabilen bir kod yazmak yazılımın geleceği için çok olumlu sonuçlar doğuracaktır.
* **Teslim ve Bakım:** Yapılan tüm testlerde yeterli görülen ve gereksinimleri karşılayacak sonuçları gösteren yazılımımızı müşteriye sunduğumuz aşamadır. Bakım aşaması ise müşteriyle bizim aramızdaki karşılıklı etkileşim sürecidir. Yazılım hep güncel tutularak ileride oluşacak yeni gereksinimlere de karşılık vermesi sağlanarak dinamizm sağlanmış olur.

**Yazılım Süreç Modelleri**

* **Gelişigüzel model:** Bu model sistematik bir yöntem içermez. Genellikle tek bir yazılımcı bütün işleri halletmeye çalıştığı için kontrolü zor bir modeldir.
* **Barok model:** 70li yıllarda kullanılan bu modelde döngü adımları doğrusal olarak hayata geçirilir. Geri dönüşleri tanımsızdır. Dinamik olmayan bir model olduğu için günümüzde kullanmamaktadır.
* **Kodla ve Düzelt:** Küçük projeler ve prototipler için kullanılan modeldir. Ön hazırlık gerektirmez, sonuca ulaşma odaklıdır. Netleşmemiş fikirlerin hayata geçirilmesiyle başlayan ve ürünün teslimine kadar kodlanmaya devam edilen modeldir. Geliştirmenin kolay olduğu kadar bakımının zor olduğu bir modeldir. Bakım maliyeti çok yüksektir.

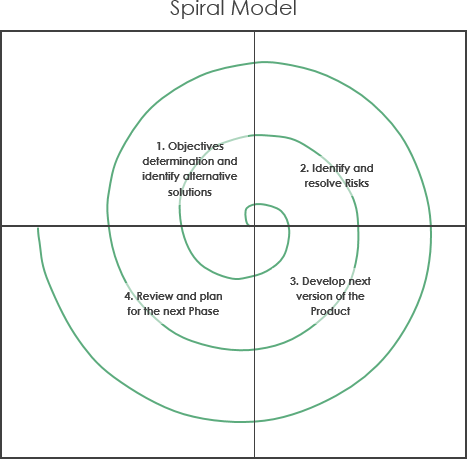


* **Şelale modeli:** Geçmişte en popüler model olarak görülen bu model günümüzde geleneksel model olarak bilinir. Sıralı olarak aşamaların tamamlanması ile olası geri dönüşlerin önüne geçilerek kaliteli bir iş ortaya çıkartılmaya çalışılır. Aşamaların en az bir kez tekrarlanması ile oluşur. Geri dönüşler zor olacağı için gereksinimlerin çok iyi belirlenmesi gerekmektedir. Belgelendirme yapılmayan aşama tamamlanmamış sayılır.



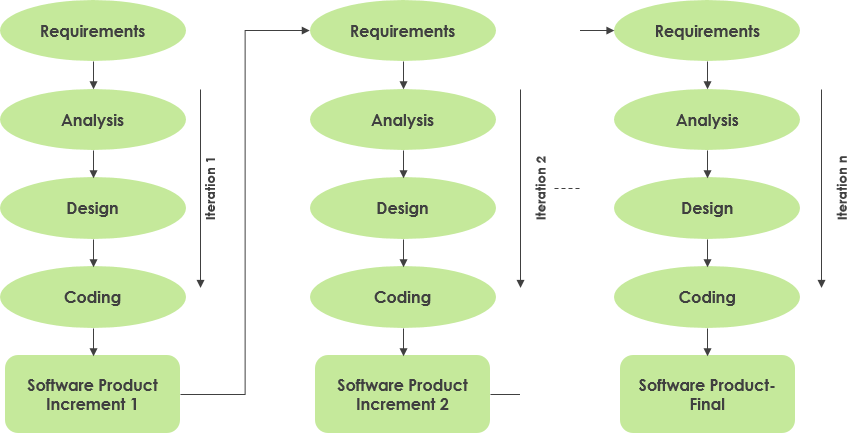
* **V modeli:** Şelale modelinin gelişmiş hali olarak ele alabileceğimiz bu yöntemin özellikleri birikimli ilerleyişi ve kontrol kolaylığıdır. Kodlama aşamasından itibaren kıvrımlı bir şekilde yukarı giden bir modeldir. Sol taraf üretimi, sağ taraf sınamayı temsil eder. Bu modelin bazı temel çıktıları şunlardır:

1. **Kullanıcı modeli:** Geliştirim süreci-kullanıcı ilişkisinin tanımı ve sistemin kabulunu sınamaya yönelin planların ortaya çıkartılması.
2. **Mimari model:** Sistemsel tasarım-alt sistem arasındaki sınama işlemleri.
3. **Gerçekleştirim modeli:** Modüllerin kodlanması ve sınanması.



* **Helezonik (Spiral) model:** Risk analizleri çok önemlidir. Prototipleme yapılır. Olası hataların erken farkındalığı sayesinde daha stabil bir iş ortaya çıkar. Her aşamasında yazılımcı ve kullanıcı arasındaki etkileşime önem verilir. Her döngü bir fazdır. Yinelemeli artımsal bir yaklaşım içerir. Bu model 4 temel aşamadan oluşur:

1. **Planlama:** Üretimi yapılacak ürün için amaç belirleme.
2. **Risk analizi:** Risk seçeneklerinin araştırılması ve belirlenmesi.
3. **Üretim**
4. **Kullanıcı değerlendirmesi**



* **Artımsal geliştirme süreç modeli:** Bu modelde aşamalar aşağıdaki sıraya göre gerçekleşir:

1. **Gereksinimlerin belirlenmesi**
2. **Gereksinimlerin artırımlara bölünmesi**
3. **Sistem mimarisinin tanımlanması**
4. **Sistem artırımlarının geliştirilmesi**
5. **Artırımların onaylanması ve birleştirilmesi**
6. **Sistemin onaylanması**

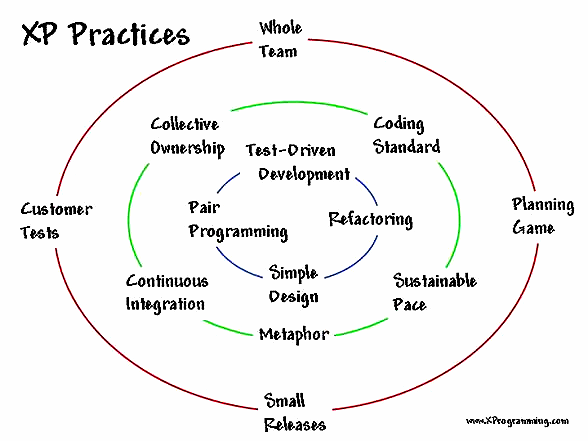
Bu modelin sürekli devamlılığı sağlanarak gelişmesi sağlanır. Yazılım son halini alana kadar üstteki döngü devam eder ve müşteriye yazılım döngüsünün her aşamasında test edilebilme imkânı sağlanmış olur.

* **Çevik modeller ve özellikleri nelerdir?**

En yaygın kullanılan birkaç modeli sıralayacak olursak:

1. **Extreme Programming (XP)**
2. **SCRUM**
3. **Agile Unified Process**
4. **Feature-Driven Development (FDD)**
5. **Test-Driven Development (TDD)**
6. **LEAN Development**
7. **Dynamic System Development Methodology (DSDM)**
8. **Microsoft Solution Framework (MSF)**

En yaygın kullanılan iki metodu inceleyecek olursak:



* **Extreme Programming (XP):** Kent Beck tarafından 90’lı yılların sonuna ortaya çıkartılan XP, grup içi iletişimi ön planda tutarak geri dönüşlerin daha verimli kullanılmasına imkân tanır. Dört temel değeri vardır:

1. **İletişim:** Eksikliği durumunda projeyi zarara uğratabilecek olan çok önemli bir faktördür. Ekip içi iletişim ne kadar iyi olursa ortaya çıkacak proje o kadar iyi olacaktır.
2. **Basitlik:** Yazılan kodun sade ve herkes tarafından anlaşılabilir olması gerekmektedir.
3. **Geri Bildirim:** Müşteriden alınan dönütler yazlımın bir sonraki adıma geçmesine yardımcı olmakta ve yazılımcının da ufkunu açmaktadır.
4. **Cesaret:** Gerektiğinde sıfırdan tekrar başlayabilecek kadar cesur olunmalı yapılacak hatalardan veya karşılaşılacak problemlerden kaçınılmamalıdır.

Takım içi anlaşma ön planda tutulduğu için anlaşmazlıklardan doğacak problemlere rastlanmaz. İletişim eksikliğini ortadan kaldırmayı amaçlar. Zorunlu işlerin yapılarak o anki gereksinimlere en uygun ve en basit çözüm yöntemi seçilmelidir. Müşteri XP sistemi içerinde sadece ürünü satın alan kişi olmaktan ziyade takımın bir üyesi konumundadır. 2-4 haftalık süreçler içerisinde sistem durumu belirlenir.

Başarısızlık durumunda en kısa zamanda telafi etmeyi amaçlar. Kolaylık ve esnekliği temel alan 12 farklı pratiği vardır:

1. **Planlama Oyunu:** Müşteri tarafından belirlenen her bir yineleme için yazılım ekibi ve müşterinin bir arada bulunduğu bir toplantıda işin yapılabileceği zaman hakkında varsayımda bulunmasıdır.
2. **Ekipte Müşteri:** Kodlama yapılırken müşterinin ekibin içinde yer alması gereksinimlere erişimi kolaylaştıracaktır.
3. **Önce Test: Test** programı olası problemleri öngörmek amacıyla yazılmalıdır.
4. **Basit Tasarım:** Anlaşılabilirliği yüksek, yönetimi ve değişimi kolay olacak şekilde tasarlanmalıdır.
5. **Çiftli Programlama:** Farklı bakış açıları ve tecrübelerin aynı proje üzerinde buluşması hataların hızlıca telafisine ve yazılımcıların gelişimine katkı sağlayacaktır.
6. **Sürekli Entegrasyon:** Sistem üzerinde yapılan değişikliklerin takibinin kolaylaşması ve olası problemlerin erken teşhis edilmesi için önemli bir aşamadır.
7. **Kısa Aralıklı Sürümler:** 2-4 haftalık periyotlarda proje güncellenerek müşteriye sunulur. Müşteri kendisini neyin beklediğini önceden deneyimleme fırsatı bulmuş olur.
8. **Yeniden Yapılandırma:** Yazılan kodu daha basit hale indirgeme ve müşteri isteklerine göre şekillendirme aşamasıdır.
9. **Ortak Kod Sahiplenme: Geliştirilen** yazılım bütün ekip üyelerince onaylanmalı ve benimsenmelidir.
10. **Metafor:** Sistemler birbirine benzetilerek yazılım geliştirmeye çalışılır.
11. **Kodlama Standardı:** Önceden tanımlı olan birtakım standartlara göre yazılım geliştirilir. Bu standartlar şunlardır:
    * + **Genel isimlendirme şekilleri**
      + **Sekmeleme, hizalama**
      + **Yorum satırları ve açıklamalar**
      + **Sınıf, metot, parametre vb. tanımlama ve kullanımı**
      + **İstisnai durum ve hata yönetimi**
      + **Dizi kullanımı**
      + **Kodlamayla ilgili öneriler vb.**
12. **Haftada 40 Saat:** Çalışma verimliliği açısından haftada 5 gün 8 saat çalışmak uygun olacaktır.



* **SCRUM:** Adını Rugby sporundaki bir hücum taktiğinden alır, bu taktikte top tüm oyuncularla birlikte karşı sahaya taşınarak atak yapılmaktadır. Jeff Sutjerland ve Ken Schawaber tarafından 1990’ların ortalarında geliştirilen çevik yazılım metodolojileriyle uygulanabilecek bir proje yönetim yaklaşımıdır. Karmaşık işlerin birden çok birime bölünmesiyle ve görev dağılımı yapılmasıyla geliştirilmesini öngörür. Gereksinimlerin tespitinde zorlanılan projeler için en uygun metodolojidir. Maksimum 30 günlük bir yineleme sürecinde günlük 15 dakikalık toplantılarla iş takibi sağlanır.
* **SCRUM’ın çalışma mantığı:**

1. **Product Backlog:** müşterinin sunduğu yüksek öncelikli gereksinim listesi
2. **Sprint Backlog:** 15-30 günlük zaman dilimine verilen addır. Genellikle bir proje birden fazla Sprint’ten oluşur.
3. **Scrum Daily Meeting:** Her gün ekip ile maksimum 30 dakikalık toplantılar düzenlenerek genel bilgi akışı sağlanır.

* **SCRUM’da temel kavramlar:**

1. **Roller**
2. **Ürün Sahibi:** Projenin iş değeri açısından geri dönüşüyle ilgilenir.
3. **Scrum Yöneticisi:** Takımın Scrum’un temel değerlerine, pratiklerine ve kurallarına bağlı kalmasını garantiler. Takımın ve organizasyonun uyum sağlamasını sağlar.
4. **Scrum Takımı:** Devamlı bir iletişim halinde olan ve hedefe yönelik hareket eden bir topluluktan oluşur.
5. **Toplantılar**
6. **Sprint Planlama:** Geniş kapsamlı gereksinim listesinin çıkartılması, dağıtım gereksinimlerinin belirlenmesi, dağıtımlar için gereksinimlerin eşleştirilmesi, takımların belirlenmesi, risk analizi yapılması, gerekli olan değişikliklerin belirlenmesi, geliştirme araçları ve altyapılarının onaylanması, maliyetlerin hesaplanması, yönetimin ve desteklerin gözden geçirilmesi ve onaylanması
7. **Sprint Gözden Geçirme:** Her sprint başlangıcında toplantılar yapılır. Toplantının ilk kısmında Ürün Gereksinim Listesi incelenerek gereksinim hedefleri ve içerikleri belirlenir. Toplantının ikinci kısmında gereksinim listesi en baştan başlanarak Sprint sonunda ortaya çıkacak olan gereksinimlerin listesini oluştururlar.
8. **Günlük Scrum Toplantısı:** Her iş gününde belirlenen saatte günlük 15 dakikalık bu toplantılar yapılır. Takımın ilerleyişini ve sorunlarını görmek için iyi bir fırsattır.
9. **Bileşenler/Araçlar**
10. **Ürün Gereksinim Dokümanı:** Yapılacak işlerin basit bir listesidir. Canlı bir dokümandır, devamlı bakım gerektirir. User story’lerden oluşur.
11. **Sprint Dokümanı:** Ürün Gereksinim Dokümanından elde edilen işleri kapsar. Amacı işlevsel ve çalışan bir son ürün elde etmektir. Sadece takım tarafından değiştirilebilir.
12. **Sprint Kalan Zaman Grafiği:** Bu grafik işlerin ne kadarının yapıldığını ve ne kadar yapılacağını karşılaştırmalı olarak gösterir.

* **SCRUM neden popüler?**

**Günümüzde artan rekabet ve talebin doğurduğu koşulları göz önüne alırsak SCRUM metodu zaman kazanmak ve sonuca ulaşmak noktalarında bize büyük yardım sağlamaktadır. Müşteriler paralarını verimli harcarken yazılımcılar ise kaliteli iletişim ortamında sorunsuzca emeklerinin karşılıklarını alabilmektedirler. İletişim düzeyinin yüksek olduğu bu çalışma ortamı sayesinde çalışanların yetkinlikleri artmakta ve işlerine motive olmaktadırlar.**

* **Hangi modeli kimler nerelerde kullanabilir?**

**Kodla ve Düzelt metodunu küçük çaplı veya tek seferlik yapılacak projelerde kullanırız.**

**Şelale modeli sürekli bir değişim gerektirmeyecek projelerde kullanırız.**

**V modeli Şelale modeli gibi gereksinimleri net olan ve az değişim isteyen projeler için uygun olacaktır.**

**Spiral modeli geniş çaplı belgelendirme gerektiren ve risk yönetimine önem verilen proje çalışmalarında kullanılmaya uygun bir modeldir.**

**Extreme Programming ve SCRUM gibi çevik modeller ise büyük çaplı ekiplerce kullanılan projeler için uygundur.**

* **Çevik modellerin ortak özellikleri nelerdir?**

1. **Müşterinin yazılımın her aşamasına tanıklık ederek canlı bir yazılım sisteminin oluşturulması.**
2. **Yeniliklere cevap verebilecek konumda olan bir yazılım yapılması.**
3. **Sağlıklı iletişim ortamında yapılan ekip çalışması.**
4. **Günlük toplantılar ve geri bildirim oturumları.**
5. **Etkin iletişim.**
6. **Yapılan her işin belgelendirilip çıktılanması.**
7. **Esneklik.**
8. **Sürdürülebilir olma.**
9. **Basitlik.**
10. **Kaliteli tasarım.**

**Kaynakça**

**Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi Yazılım Mühendisliği Temelleri Dersi Hafta 2–3 Slaytları**

[**https://medium.com/@denizkilinc/yazılım-yaşam-döngüsü-temel-aşamaları-software-development-life-cycle-core-processes-197a4b503696**](https://medium.com/@denizkilinc/yazılım-yaşam-döngüsü-temel-aşamaları-software-development-life-cycle-core-processes-197a4b503696)

[**https://medium.com/@yunus.selcuk/yazılım-yaşam-döngüsü-ve-scrum-2770a183dd7d**](https://medium.com/@yunus.selcuk/yazılım-yaşam-döngüsü-ve-scrum-2770a183dd7d)

[**https://medium.com/@solobasay/yazılım-yaşam-döngüsü-modellemeleri-857af10373bb**](https://medium.com/@solobasay/yazılım-yaşam-döngüsü-modellemeleri-857af10373bb)

[**https://hayririzacimen.medium.com/yazılım-yaşam-döngüsü-ve-süreç-modelleri-70fdfb2f8f77**](https://hayririzacimen.medium.com/yazılım-yaşam-döngüsü-ve-süreç-modelleri-70fdfb2f8f77)

[**https://mustafaserdarkonca.medium.com/yazılım-yaşam-döngü-modellerini-açıklamadan-önce-yazılıma-ve-yazılımın-nasıl-bir-gelişim-geçirerek-2f724cd8c868**](https://mustafaserdarkonca.medium.com/yazılım-yaşam-döngü-modellerini-açıklamadan-önce-yazılıma-ve-yazılımın-nasıl-bir-gelişim-geçirerek-2f724cd8c868)