**Yazılım Yaşam Döngüsü (SDLC)¹ ve Modelleri**

Proje yönetimi, uygulanması gereken adımlar ve uyulması gereken kurallardan kaynaklı karmaşık bir süreçtir. Yazılım projelerinin yönetiminde de benzer şekilde belirli adımlar ve karmaşıklıklar vardır. Bu temel adımlarından bahsetmek gerekirse; Planlama, Analiz, Tasarım, Gerçekleştirme (Kodlama ve Test), Teslim ve Bakım olarak sıralayabiliriz.

Karşılaştığımız veya karşılaşabileceğimiz sorunlar, Müşteri istekleri, Maliyet gibi dış etkenlerden kaynaklı döngü içerisinde herhangi bir aşamada belirli adımlara dönmek ve tekrar ilerlemek gerekebilir. Süreç iyi yönetilmediği takdirde proje sekteye uğrayacak daha üst seviyelerde ise çöpe gidecektir. Bu gibi sorunlarla karşılaşmayıp sürecin yönetilmesini kolaylaştırmak amacıyla da bazı modeller öne sürülmüş ve belli durumlarda kullanılmaya başlamıştır.

En eski, en ünlü ve de geçmişin en popüler yazılım geliştirme modellerinden biri olan **Şelale Modeli (Waterfall Model)²** ile başlamak gerekirse bu model, her aşamanın en az bir kez tekrarlanması ve bir aşama tamamlanmadan diğer aşamaya geçilmemesi üzerine kurulmuştur. Şelale modelinde Analiz ve tasarım aşaması çok uzun sürse de problemlerin iyi belirlenmesi sayesinde “gerçekleştirme” aşamasında daha az hata ile hızlı bir şekilde ilerleme sağlar. *Analiz ve tasarımın çok önemli olduğu, hata yapma lüksünün olmadığı veya sorunun çok iyi tanımlandığı projeler için uygun bir modeldir.*

Şelale Modelinin temel özelliklerinden bahsetmek gerekirse;

* Tamamlanan her adımın sonunda bir doküman oluşturulur yani doküman ile desteklenir,
* İstek ve gereksinimler başlangıç safhasında belirlenir yapısı gereği daha sonradan bir ekleme veya çıkarma yapılamaz,
* Bir adım eksiksiz olarak tamamlanmadan sıradaki adıma geçilmez,

Modelin getirdiği problemler ve kullanılmama sebepleri:

* Adımlar arasındaki keskin çizgi yapılan hataların telafisini oldukça güç ve maliyetli kılabilir.
* Değişikliğe açık bir model değildir. Ne istediğini bilmeyen müşterilerde gereksinimlerin değişmesi yazılımın yapısı baştan sona değiştirebilir ki bu durum da maliyet ve zaman bakımından yazılımcıyı zora sokacaktır.
* Üretilen yazılımın kullanılır hale gelmesi uzun zaman alabilir ve kullanıcı sürecin içerisine dahil edilemez bu durum proje tamamlandıktan sonra geri dönüşlere sebep olabilir.

En sık kullanılan modellerden birisi olup aynı Şelale Modeli gibi doğrusal modellerden birisi olan **V-Model (V-shaped Model)³** ise işleri biraz daha ileri taşıyıp şelale modelinin kontrol adımını daha düzenli hale getirmiştir. V-Model ismini her adımın kendi test aşamasıyla eşleştirilmesiyle oluşan “V” harfine benzerliğinden almıştır. Her adımın kontrol edilerek ilerlenmesi bu modelin en büyük avantajlarından birisidir. *Belirsizliğin az iş tanımlarının belirgin olduğu projeler için uygun bir modeldir.*

V-Model'in temel çıktıları üçe ayrılır ve bunlar Kullanıcı Modeli, Mimari Model, Gerçekleştirim modeli olarak sıralanır.

* Kullanıcı Modeli: Projenin kullanıcıyla olan ilişkilerini tanımlayan ve bunlara bağlı sınama işlemleridir.
* Mimari Model: Sistem tasarımı ve oluşacak tüm sistemlerin sınama işlemleridir.
* Gerçekleştirim modeli: Yazılımın kodlanması ve test edilmesine ilişkin fonksiyonlardır.

Modelin getirdiği problemler ve kullanılmama sebeplerine bakarsak

* Şelale Modelinin test kısmını standartlaştırmak dışında pek bir yenilik eklemediği için Şelale Modeliyle benzer sorunlarla karşılaşıyoruz.
* Kesin kurallara bağlı olduğu için esnekliği düşüktür.
* Herhangi bir aşamada gereksinimlerin değişmesi durumunda test belgelerinde de ciddi değişiklikler yapılması gerekir.

Sıradaki modelimiz olan **Helezonik Model (Spiral Model)⁴** risk analizi ve prototip üretme üzerine kurulmuştur. Bu modelde her döngüden önce içinde bulunduğu adımın risk analizi yapılıp o adımın planlanmış prototipi geliştirilir. Döngü her sona erdiğinde yeniden planlama yapılarak hedef, alternatif ve kısıtlamalar belirlenir. *Helezonik* *Model önceden geliştirilmiş yazılım ürünlerinin yeniden kullanıldığı projeler için çok uygundur.*

Helezonik yazılım geliştirme modeli 4 alt başlığı içerir ve bunlar Planlama, Risk Analizi, Üretim ve Kullanıcı Değerlendirmeleri olarak sıralanabilir.

* Planlama: Üretilecek ara ürünün adımlarının planlanması; amacını, kısıtlamalarını ve alternatiflerini belirleyip bir önceki adımda üretilen ürün ile bütünleştirilmesini kapsar.
* Risk Analizi: Planlama aşamasında belirlenen alternatiflerin değerlendirilmesi ve risk analizini kapsar.
* Üretim: Planlanan ara ürünün geliştirildiği ve üretildiği bölümdür.
* Kullanıcı Değerlendirmesi: Üretilen ara ürünün kullanıcıya sunulmasını ve kullanıcılardan gelen değerlendirmeleri kapsar.

Helezonik Model tercih edilmesindeki en büyük etkenler

* Her döngünün başında risk analizi yapıldığı için projenin ihtiyacı olan zaman ve maliyetin daha kolay bir şekilde tahmin edilmesi.
* Üretilen ara ürünler ile kullanıcı sürece erken dahil edildiğinden proje bitiminde geri dönüşler az ve müşteri memnuniyeti daha fazla olur.
* Gerek proje sahibi gerekse proje yöneticileri elle tutulur çalışan yazılımları gördüğü için süreci daha rahat takip edebilir.
* Proje yazılımının kodlanması ve test edilmesi daha erken başlar.

Modelin getirdiği problemler ve kullanılmama sebepleri:

* Helezonik Model yapısı gereği küçük projelere kullanılmaya uygun değildir.
* Döngülerin genelinde sürekli bir risk değerlendirmesi olduğundan bu konuda tecrübeli veya uzman kişilere ihtiyaç duyulur.
* Çok fazla ara aşama bulunduğundan belgeleme işlemi fazladır.

Müşterinin istediği sistemde her modülün önemi aynı seviyede olmayabilir. Bu gibi durumlarda kullanılmak üzere “Böl ve Yönet” yaklaşımı ile **Artımsal Geliştirme Süreç Modeli (Incremental Development**)**⁵** ortaya çıkmıştır. Modelin amacı müşterinin belirttiği gereksinimleri önceliklendirerek belirlenen gereksinimleri erken teslime sunmaktır yani sistemi tek seferde sunmaktansa parçalara bölüp üzerlerine ekleme yapmak esastır. Sisteme yapılan her ekleme yeni bir sürüm olarak adlandırılır ve her yeni sürüm giderek artan sayıda işlev barındırır. *Uzun zaman alabilecek ve sistemin eksik işlevsellikte çalışabileceği projeler için uygun bir modeldir.*

Artımsal Geliştirme Süreç Modelinin tercih edilmesindeki en büyük etkenler:

* Hızlı ve erkenden kullanılabilir durumda yazılım oluşturulabilir.
* Gereksinimleri değiştirmek daha az maliyetlidir.
* Müşteriden her sürüm için değerlendirmeler alınabilir ve bu durum projenin geri dönüşlerini azaltır.
* Projenin tamamının başarısız olma ihtimalini azaltır.
* Riski yönetmek daha kolaydır ve yazılımcıya bir hatayla karşılaşıldığında geri dönebileceği sürümler sunar.

Modelin getirdiği problemler ve kullanılmama sebepleri:

* Yapılacak eklemeleri tanımlamak için tüm sistemin düzgün tanımlanması ve iyi tasarlanması gerekir.
* Süreci yönetmek için deneyimli personel gerekir.

Henüz tam olarak belli olmayan bir ürün fikriyle kodlamaya başlayıp ürün istediğiniz seviyeye geldiğinde biten **Kodla ve Düzelt Modeli (Code and Fix)⁶** yazılım geliştirmenin en kolay yolu ancak yerine göre en pahalısıdır. Bu model genel olarak denetimsizdir yani planlaması yoktur ve bitiş zamanı belirsizdir. Sisteme ait bir dokümantasyon bulunmadığı için hataların düzeltilmesi zordur ve bakım yapılabilirliği azdır. Belli bir noktadan sonra emeklilik (retirement) safhasına girer. *Küçük projeler ya da kısa süreli prototipler için uygundur ve öğrenciler ve bireysel geliştiriciler tarafından kullanılabilir.*

Kodla ve Düzelt Modelinin tercih edilmesindeki en büyük etkenler:

* Herhangi bir planlama gerekmez.
* Uzmanlık gerektirmez yani herkes bu modeli kullanabilir.
* Program adımları çabuk geçilir.
* Bireysel çalışmalara uygundur.

Modelin getirdiği problemler ve kullanılmama sebepleri:

* Bitiş süresi belli değildir
* Hataların bulunması ve düzeltilmesi zordur
* Doğaçlama çalışıldığı için esnek projeler sunmaz yani sonradan değiştirilmesi veya üzerine ekleme yapılması zordur.
* Kontrollü değildir, kaynak planlaması yoktur

**Çevik Yazılım Geliştirme:**

Verimliliği yüksek, esnek, hata oranı düşük, hızlı ve ucuz çözümler. Bu maddeler çevik yazılımın genel özellikleridir. “Heavyweight” olarak bilinen eski ve yavaş sistemlere tepki olarak doğan bu geliştirme metotları projenin ölçeği ne olursa olsun onu küçük yinelemelere ayırır ve ayrılan her bir parçaya yeni bir proje gözüyle bakar. “İlerlemenin en iyi göstergesi, çalışan bir yazılımdır” prensibini esas alarak ayrılan her bir yinelemenin sonunda müşteriye projede ne kadar ilerlendiğine dair bilgi verir ve bir yazılım sunar. Çevik Yazılımın öncelikleri şunlardır:

Süreçler ve araçlar yerine Bireyler ve Etkileşimler, Kapsamlı Belgeler yerine Çalışan Yazılım, Sözleşme Görüşmeleri yerine Müşteri İlişkileri, Plan İzleme yerine Değişikliğe Açıklık

* Projenin hızı ekibin sürekli iletişimiyle her bir yinelemenin 2 ila 4 hafta arasında sürmesi planlanarak sağlanır.
* Projenin küçük parçalara ayrılmış olması da geriye dönük hataların düzeltilmesi kolaylaştırır ve esnekliği sağlar.
* Ürünün ve yazılımcının sürekli müşteriyle muhatap olmasından kaynaklı müşteri memnuniyeti fazla ve hata oranı düşüktür.

Her süreçte olduğu gibi Çevik Yazılım Geliştirmenin de dezavantajları vardır.

* Daha rahat ve sürekli iletişim gerektirdiği için kurumsal şirketlerde uygulaması zordur bu sebeple çoğunlukla bağımsız şirketler tarafından tercih edilir.
* İhtiyaçların sürekli değişmesinden kaynaklı yoğun bir çalışmaya temposuna ihtiyaç duyulur.

**Scrum⁷:**

Çevik yazılım geliştirmede en popüler metodolojilerden birisi olan Scrum temelde kompleks yazılımlarda tercih edilmektedir. Bunu yaparken bütünü parçalayan, tekrara dayalı bir yol izler. Tüm çevik yazılım geliştirme yöntemlerinde olduğu gibi düzenli geri bildirimler ve planlamalar ile hedefe doğru yol alır.Scrum için iletişim ve takım çalışması olmazsa olmazdır ve bu durum esnekliğinin temelinde yer alır.

3 temel prensip içerisinde ele alınır. Projenin ilerleyişi, sorunlar ve gelişmeler herkes tarafından görülebildiği için “Şeffaf”. Projenin ilerleyişi sürekli olarak kontrol edildiğinden “Denetimli”. Esnek yapısından kaynaklı değişikliklere uyum sağlayabilen yani “uyarlanabilir”dir.

**Scrum Çalışma Mantığı:**

Product Backlog: Müşteriyle beraber belirlenen öncelikli ve yüksek seviyeli gereksinimlerdir.

Sprint Backlog: Genelde 15-30 gün (Time-Boxed) olan proje zaman dilimidir. Her proje birden fazla Sprint bulundurur.

Scrum Daily Meeting: Her gün geliştirme takımıyla beraber maksimin 30 dakika ve genelde ayakta yapılan günlük toplantılardır. Amaç süreç hakkında genel bilgi almak ve takım arasındaki iletişimi güçlü tutmaktır. İçerikler “ Dün ne yaptın? Bugün ne yapacaksın? Hangi engellerle karşılaştın? “ gibi sorulardır

Scrumda 3 temel kavram vardır ve bunlar Roller, Toplantılar ve Bileşenler/Araçlar’dır

1. Roller (Roles)

* Ürün Sahibi (Product Owner): Geliştirme takımı ve müşteri arasındaki köprüdür. Projenin iş değeri bakımından geri dönüşü ile sorumludur.
* Scrum Yöneticisi (Scrum Master): Scrum kuralları, teorileri ve pratiklerinin uygulanmasını garanti altına alır. Takım yöneticisi değildir.
* Scrum Takımı (Scrum Team): Kendi kendini yöneten, devamlı iletişim halinde olup tek bir hedefe ulaşmak için mücadele eden üyelerden oluşur. Sprint backlog’u oluşturmak da Scrum takımının görevidir.

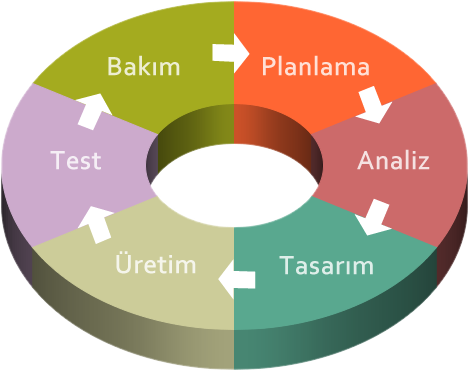
1. Toplantılar (Meeting)

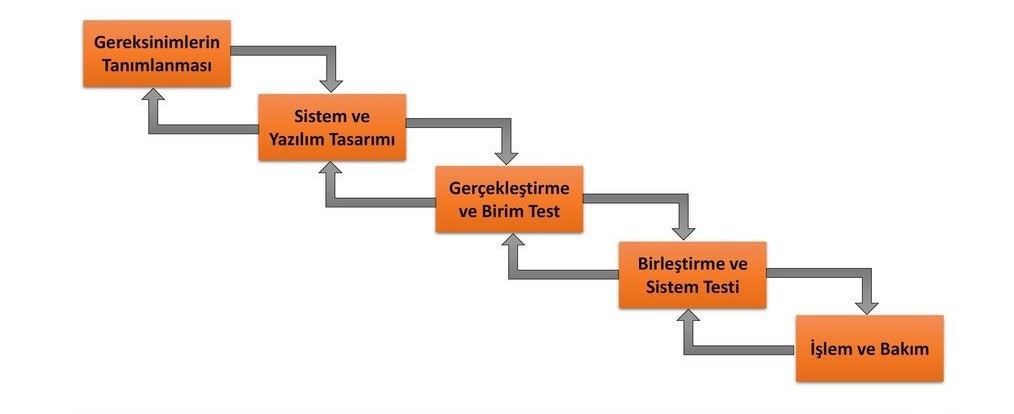
* Sprint (Koşu) Planlama (Sprint Planning): Product backlog ile belirtilen gereksinimler bu toplantıda scrum takımı tarafından küçük görevlere ayrılır ve bu takımdaki herkes bu görevleri kendi arasında paylaşır. Bu toplantıya ürün sahibi ve scrum yöneticisi de katılır. Toplantının diğer amaçlarından birisi de gereksinim, risk analizi, maliyet gibi konuların da konuşulmasıdır. Toplantının sonunda ürün sahibine sunulmak üzere en az bir sprint belirlenir.
* Günlük Scrum Toplantısı (Daily Scrum Meeting): Günlük olarak yapılan kısa toplantılardır ve tüm takım katılır. Takımın ilerleyişini görmek için uygun toplantılardır.
* Sprint (Koşu) Gözden Geçirme (Sprint Review): Her sprint (koşu) sonunda yapılır ortaya çıkan iç gözden geçirilir ve ürün değerlendirilir. Amaç Bir hata fark edilirse düzeltmektir.

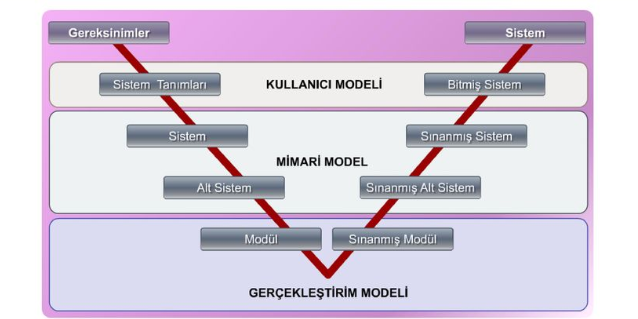
1. Bileşenler/Araçlar (Artifacts)

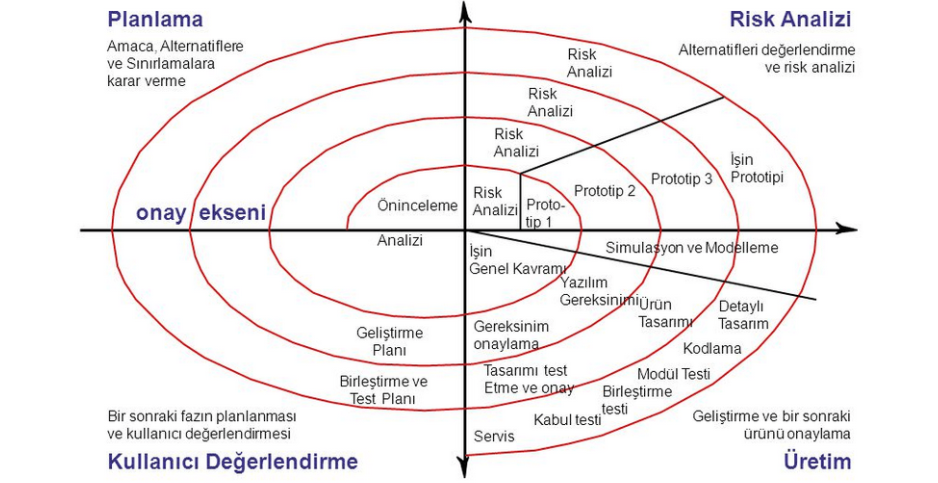
* Ürün Gereksinim Dokümanı (Product Backlog): Proje boyunca yapılması gereken işlerin bir listesidir. Sürekli olarak bakım gerektirir yani yeni elemanlar eklenebilir veya çıkartılabilir. Ürün geliştirme dokümanları genelde kullanıcı gözünden yazılır.
* Sprint (Koşu) Dokümanı: Mevcut sprint için product backlog’dan elde edilmiş iş ve görevleri kapsar.
* Sprint Kalan Zaman Grafiği (Burndown Chart): Kalan zaman grafiği sprint boyunca planlana işlerin ne kadarının yapıldığını ve ne kadarının yapılması gerektiğini karşılaştırabilmemizi sağlar. Takım elemanlarının günlük olarak ne kadarlık bir iş gerçekleştirdiklerini girmesi ile doldurulur.

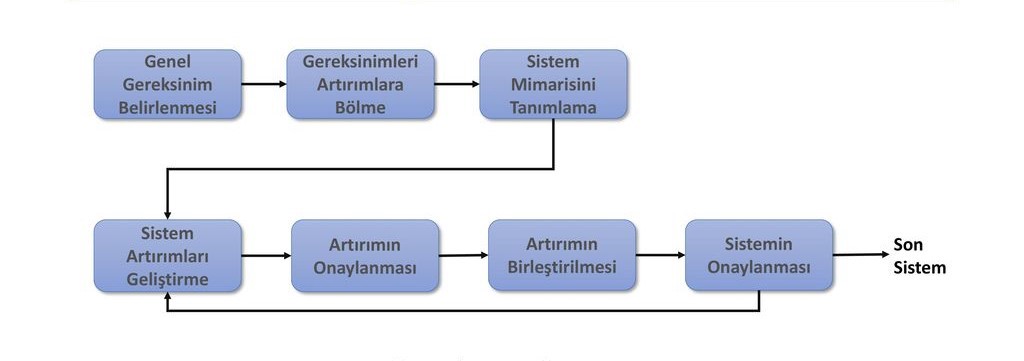
**Yazılım Yaşam Döngüsü (SDLC):**

**(şekil - 1)**

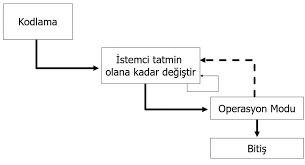
**Şelale Modeli (Waterfall Model): (şekil - 2)**

**V-Model (V-shaped Model): (şekil - 3)**

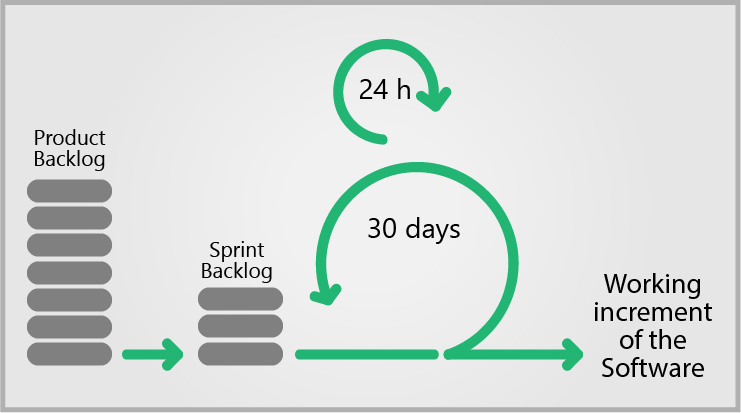
**Helezonik Model (Spiral Model): (şekil - 4)**

**Artımsal Geliştirme Süreç Modeli (Incremental Development**):** (şekil - 5)**

**Kodla ve Düzelt Modeli (Code and Fix):**

** (şekil - 6)**

**Scrum:**



**(şekil - 7)**

**Kaynakça:**

[1] <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>

[2] <https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8>

[3] <https://fikirjeneratoru.com/yazilim-proje-yonetimi-yontemleri/>

[4] <https://www.yazilimtestmerkezi.com/post/yazilim-gelistirme-modeli-nedir-cesitleri-nelerdir>

[5] <https://www.codex.com.tr/yazilim-gelistirme-modelleri>

[6] <https://hayririzacimen.medium.com/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-ve-s%C3%BCre%C3%A7-modelleri-70fdfb2f8f77#:~:text=Kodla%20ve%20D%C3%BCzelt%20(Code%20and,i%C3%A7in%20herkes%20bu%20modeli%20kullanabilir>.

[7] <https://koraypeker.com/2018/03/02/sdlc-waterfall-model/>

[8] <http://www.aspmvcnet.com/tr/m/yazilim-muhendisligi/helezonik-tasarim-spiral-tasarim.html>

[9] <https://tr.csstricks.net/8222532-spiral-model-when-to-use-advantages-and-disadvantages>

[10] <https://medium.com/@berkecanbay/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-d5234f64c758>