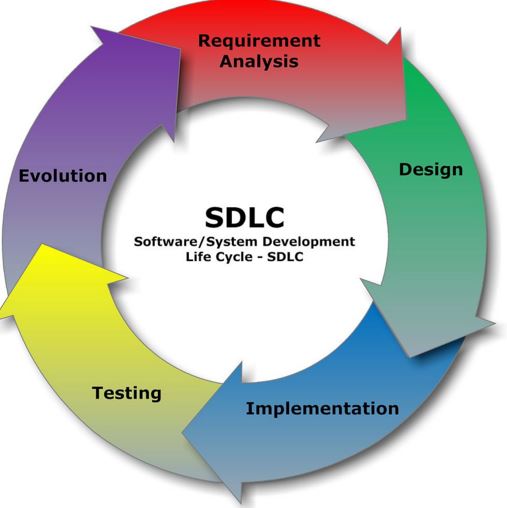
**Yazılım Yaşam Döngüsü Modelleri (Software Development Life Cycle Models)**

**Yazılım Yaşam Döngüsü**

Yazılım yaşam döngüsü,yazılım uygulamarı olşturmak için bazı aşamaların gerçekleştirilmesidir.Bu aşamalar genellikle şöyledir:Planlama,Gereksinimlerin analizi,Tasarım,Gerçekleştirme,Test etme,Bakım

**Planlama(Planning):** Planlama aşamasında, proje lideri projeyi enine boyuna ele alır.Bu aşama,maliyetleri hesaplama,zaman çizelgesi oluşturma,hedef belirleme ve liderlik yapısını oluşturmayı içerir.Planlama aşamasında müşteriden gelen geri bildirimleri de içerebilir.Potansiyel müşterilerden,konunun uzmanlarından,satış temsilcilerinden geri bildirimler alınmaya çalışılır.Planlama aşamasında,uygulamanın kapsamı ve amacı tanımlanmalıdır.Proje boyunca izlenecek rota belirlenir.Belirlenen rota sonucu takımın yazılımı tamamlamasını etkiler.Ayrıca belirlenen rota ,projeinin orijinal amacının dışına çıkmaması için bazı sınırlar belirler.

**Gereksinimlerin Tanımlanması(Define Requirements):** Gereksinimlerin tanımlanmasında uygulamanın ne yapması gerektiğini ve gereksinimlerin belirlenmesi asıl amaçlardandır.Bu nedenle planlamanın bir parçası olarak kabul edilir.Gereksinim belirlenmesine örnek verecek olursak,günlük hayatımızda kullandığımız sosyal medya uygulamalarının karşılıklı bağlantı kurma becerileri ve arama özelliği olması gerekebilir.Ayrıca bu aşamada projeyi oluşturmak için gereken kaynaklar da tanımlanır.

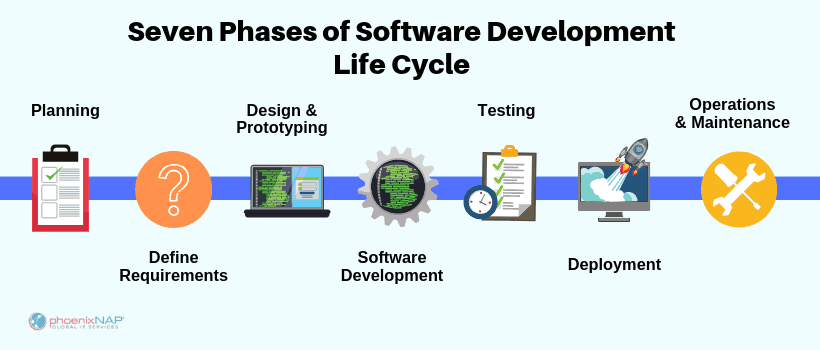
**Tasarım(Design):**Tasarım aşaması,yazılımın çalışma şeklini modeller.Progalama dili,genel tasarımı ve şablonu mimarı yapısını belirler.Müşterilerin yazılımla etkileşime geçebilme yolları ve yazılıma nasıl tepki verecekleri tanımlar.Programlama kısmında yalnızca programa dili değil uygulamada sorunları çözme yöntemlerini de içerir.

**Yazılım Geliştirme(Software Development):**Yazılım geliştirme aşaması asıl programın asıl yazıldığı aşamadır.Küçük bir proje bir kişiyle bile yazılabilirken ,büyük bir proje için koca bir ekip gerekebilir.Büyük proje böylece ekiplere paylaştırılabilir.Bu aşamada erişim kontrolü ve kaynak kodu yönetimi(Source Code Management) uygulaması kullanılabilir.Bu uygulamalar koddaki değişiklerin izlenmesi sağlar.Ayrıca farklı ekiplerin arasında uyum sağlamaya yardımcı olur.Bu aşamada hataları bulmak ve düzeltmek çok önemlidir.

**Test Etme(Testing):**Uygulamayı müşteriye sunmadan önce test etmek çok önemlidir.Test etme aşamasında uygulamanın her işlevinin doğru çalıştığından emin olunmalıdır.Uygulamanın farklı bölümlerinin birlikte sorunsuz bir şekilde çalıştığı test edilmelidir.Gecikmeleri,takılmaları azaltmak için performans testi yapılır.Test aşaması, kullanıcınnın karşılaştığı aksaklıkların azalmasına yardımcı olur.Bu sayede yüksek bir kullanıcı memnuniyeti sağlar.

**Dağıtım(Deployment):** Dağıtım aşamasında, uygulama kullanıcılara sunulur.Dağıtım aşaması da karmaşık olabilir.Şirket çapında bir veritabanını yeni geliştirilmiş bir uygulamaya yükseltmek buna bir örnektir. Veritabanında kullanılan birkaç başka sistem olduğundan, yükseltmeyi entegre etmek daha fazla zaman ve çaba gerektirebilir.

**Bakım(Maintenance):**Bu aşamada, geliştirme kısmı neredeyse bitmiştir.Uygulama sahada kullanılmaktadır. Bu aşamada, kullanıcılar test sırasında bulunmayan hataları bulurlar. Yeni geliştirme döngüsü, kısmında ortaya çıkabilecek bu hataların çözülmesi gerekir. Hata düzeltmelerine ek olarak, yinelemeli geliştirme gibi modeller gelecekteki sürümlerde ek özellikler planlar. Her yeni sürüm için yeni bir geliştirme döngüsü başlatılabilir.

****

**Yazılım Geliştirme Yaşam Döngü Modelleri**

**(Software Development Life Cycle Models)**

**➞Şelale Modeli**

**➞V Şeklinde Model**

**➞Evrimsel Prototipleme Modeli**

**➞Spiral Yöntem (SDM)**

**➞Yinelemeli ve Artımlı Yöntem**

**➞Çevik geliştirme**

**Şelale Modeli(Waterfall Model)**

****

Şelale modeli,klasik geliştirme yöntemidir.Şelale modeli,doğursal olarak ilerleyen sıralı akıştır.Yazılım uygulama aşamalarında sürekli olarak aşağı doğru (bir şelale gibi) aktığı görülür. Her aşama tamamlandığında, proje bir sonraki aşamaya geçer. Bu, geliştirme sürecindeki herhangi bir aşamanın yalnızca önceki aşama tamamlandığında başlayacağı anlamına gelir. Şelale modelinin bir avantajı, devam etmeden önce her aşamanın değerlendirilebilmesidir. Ancak hız sınırlıdır, çünkü bir aşamanın diğerinin başlayabilmesi için bitmesi gerekir. Ayrıca şelale modeli, gereksinimdeki değişiklikleri ele almak için önceki aşamaya geri dönme sürecini tanımlamaz. Şelale modeli, yazılım geliştirme için kullanılan en eski ve en çok bilinen modelidir.

**Avantajları** **Dezavantajlar**ı

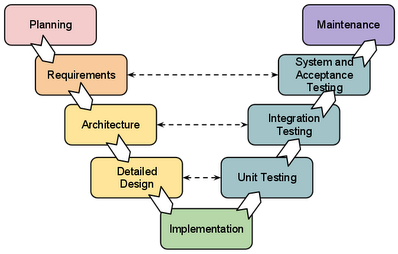
•Kullanıcılara açıklaması kolaydır. •Bittikten sonra herhangi bir aşamaya dönmek çok zordur.

•Aşamalar ve aktiviteler iyi tanımlanmıştır. •Maliyetli ve daha fazla zaman gerektirir.

•Projenin planlanmasına yardımcı olur. •Kapsamı ayarlamak zor ve pahalıdır.

•Her aşamanın çıktıları belirlidir.

**V-Modeli (V-Shaped Model)**



Şelale modelinin bir uzantısıdır, Doğrusal bir şekilde aşağı inmek yerine, tipik V şeklini oluşturmak için uygulama ve kodlama aşamasından sonra işlem adımları yukarı doğru bükülür. V şeklindeki model ile şelale modeli arasındaki en büyük fark, V şeklindeki modeldeki erken test planlamasıdır.

**Avantajları** **Dezavantajlar**ı

• Basit ve kullanımı kolaydır. •Şelale modeli gibi çok esnek değildir.

• Her aşamanın belirli çıktıları vardır. •Kapsamı ayarlamak zor ve pahalıdır.

• Gereksinimlerin kolayca anlaşıldığı yerlerde •Ayrıntılı bir plana ek olarak maliyetli ve daha fazla

iyi çalışır. zaman gerektirir

• Yaşam döngüsünün başlarında test planlarının • Model, test aşamaları sırasında bulunan

geliştirilmesi nedeniyle şelale modeline göre sorunlar için net bir yol sağlamaz.

daha yüksek başarı şansı vardır. • Yazılım, uygulama aşamasında geliştirildiğinden,

•Ürün geliştirmenin ilk aşamalarında ürünün yazılımın erken prototipleri üretilmez.

doğrulanması ve onaylanması vardır.

**Prototip Oluşturma Modeli**

**(Prototyping Model)**

Yazılım uygulamalarının prototiplerini geliştirilmekte olan yazılım programının eksik sürümlerini oluşturma faaliyetini ifade eder. Yazılım geliştirmede meydana gelebilecek bir faaliyettir ve geliştirme ekibinin müşteri gereksinimlerini yanlış anlama boşluğunu sınırlandırmak için yazılımın bazı bileşenlerini görselleştirmek için kullanılır. Bu aynı zamanda şelale modelinde meydana gelebilecek yinelemeleri de azaltacak ve şelale modelinin esnekliği nedeniyle uygulanması zor olacaktır. Bu nedenle, nihai prototip geliştirildiğinde, ihtiyacın dondurulmuş olduğu kabul edilir.

****

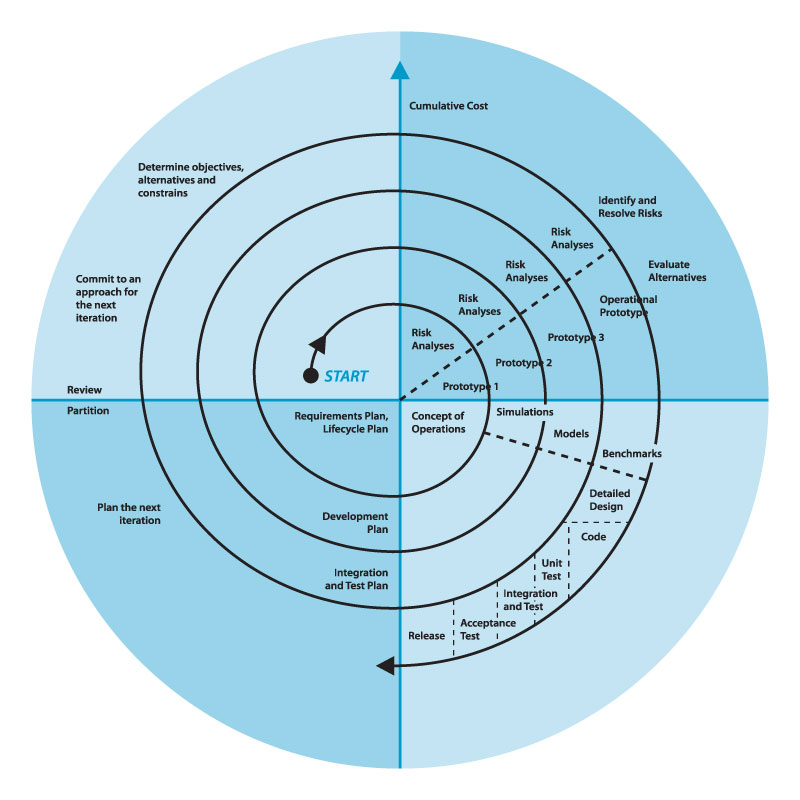
**Avantajlar**

•Daha az zaman ve maliyet, ancak geliştirici prototipleri geliştirirken zaman kaybederse bu bir dezavantaj olabilir. •Geliştirilmiş ve artan kullanıcı katılımı sağlar.

**Dezavantajları**

•Analiz kısmında yetersizdir. •Prototip ve bitmiş sistem konusunda kullanıcı kafa karışıklığı yaşayabilir. •Geliştiricinin kullanıcı hedeflerini yanlış anlama ihtimali yüksektir. •Prototipin geliştirme süresi fazladır. •Prototipleri uygulamak maliyetlidir

**Spiral Model**

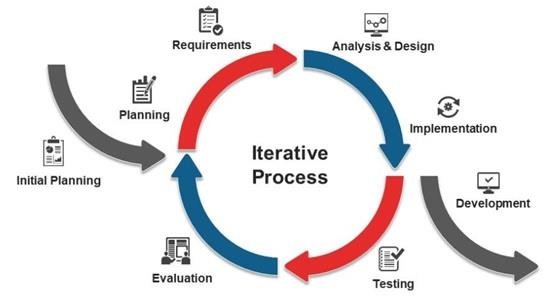


Yukarıdan aşağıya ve aşağıdan yukarıya konseptlerin avantajlarını birleştirme çabasıyla, hem tasarım hem de aşamalı prototip oluşturma unsurlarını birleştirilir. Bu geliştirme modeli, prototip oluşturma modelinin ve şelale modelinin özelliklerini birleştirir. Spiral model, büyük, pahalı ve karmaşık projeler için tercih edilmektedir. Bu model, planlama, risk değerlendirmesi ve prototip ve simülasyonların oluşturulmasıyla ayrılmış, esasen şelale modeliyle aynı aşamaların çoğunu kullanır.

**Avantajlar Dezavantajları**

•İş ilerledikçe tahminler (yani bütçe, program vb.) • Nihai ürüne ulaşmak için yüksek maliyet ve zaman daha gerçekçi hale gelir çünkü önemli konular harcanır. daha erken keşfedilir. • Riskleri ve varsayımları değerlendirmek için özel • Geliştiriciler erken katılım sağlar. becerilere ihtiyaç duyar. •Riskleri yönetir ve sistemi aşamalar halinde geliştirir. •Son derece özelleştirilmiş, sınırlayıcı yeniden • kullanılabilirlik özelliğine sahiptir.

**Yinelemeli - Artımlı Model Iterative - Incremental Model)**



Yinelemeli modelleme, şelale modelinin zaaflarının üstesinden gelmek için geliştirilmiştir. İlk olarak planlamayla başlar ve aradaki döngüsel etkileşimlerle dağıtımla biter. Bu yöntemin arkasındaki temel fikir, tekrarlanan döngülerle (yinelemeli) ve bir seferde daha küçük bölümler halinde (artımlı) bir sistem geliştirerek, yazılım geliştiricilerin sistemin önceki parçalarının veya sürümlerinin geliştirilmesi sırasında öğrenilenlerden yararlanmalarına izin vermektir. Mini şelalelerden veya mini V-şekilli modelden oluşabilir.

**Avantajlar**

•Geliştirme yaşam döngüsünün başlarında iş değeri üretir.

•Kaynakların daha iyi kullanılması.

•Artımlar arasında bazı değişiklik taleplerini karşılayabilir.

•Müşteri değerine odaklanmıştır.

• Sorunlarını ve değişiklikleri daha erken tespit edebiliriz.

**Dezavantajları**

•Ağır dokümantasyon gerektirir.

•Tanımlanmış bir süreçler dizisini izler.

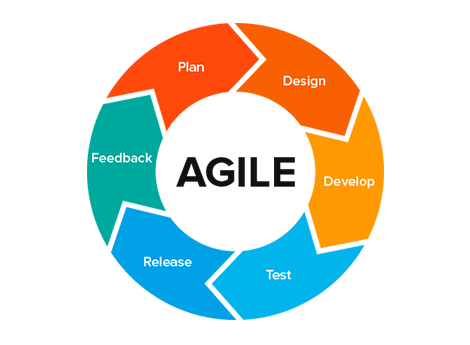
•Fonksiyon ve özellik bağımlılıklarına göre artışları tanımlar.

•Doğrusal yaklaşımlardan daha fazla müşteri katılımı gerektirir.

•İşlevlerin ve özelliklerin bölümlenmesi sorunlu olabilir.

•Geliştirme ve proje planlaması sırasın dikkate alınmazsa, yinelemeler arasındaki entegrasyon bir sorun olabilir.

**Çevik Model(Agile Model)**



Gereksinimlerin ve çözümlerin işlevler arası ekipler arasındaki işbirliği yoluyla geliştiği yinelemeli ve artımlı geliştirmeye dayanır.

**Scrum Agile Model**

Her tür projede kullanılabilir, ancak müşterinin daha fazla katılımcı ve etkileşimli olması gerekir. Ayrıca, müşterinin bazı işlevsel gereksinimlerini üç haftadan daha kısa sürede hazırlaması gerektiğinde ve gereksinimler yeterince net olmadığında kullanabiliriz. Bu, yazılım için daha değerli ve işlenebilir parçaların erken aşamada olmasını sağlayacak ve aynı zamanda müşteri memnuniyetini artıracaktır.



**Avantajlar Dezavantajları •**Bazı sistem özelliklerinden yararlanmak için **•**Ölçeklenebilirlik yönünden diğer modellere gereken süreyi azaltır. Kıyasla dezavantajlıdır. •Müşteri temsilcisinden yüz yüze iletişim ve **•** Müşterinin, kullanıcı ihtiyaçlarını ifade etme sürekli girdiler sağlarlar,bu sayede tahminlere yer bırakmaz. yeteneği ve işbirliğine bağlıdır. •Sonuç,genellikle mümkün olan en kısa sürede yüksek **•**Ekip için özel becerilere ihtiyaç vardır. kaliteli yazılım ve memnun müşteridir.

**Scrum Neden Bu Kadar Popüler ? Scrum Neden Bu Kadar Başarılı ?**

Bu sorulara doğrudan cevap vermek gerekirse, Scrum'ın basit, anlaşılır ve uygulaması kolay olmasıdır. Sadece doğal özellikleri nedeniyle değil, aynı zamanda iyi belgelenmiş kullanım durumlarına sahip olduğu için başarılıdır. Elbette Scrum'ın başarısının bir başka iyi nedeni de işe yaramasıdır.Ayrıca Scrum üretkenliğin artmasını da sağlar.

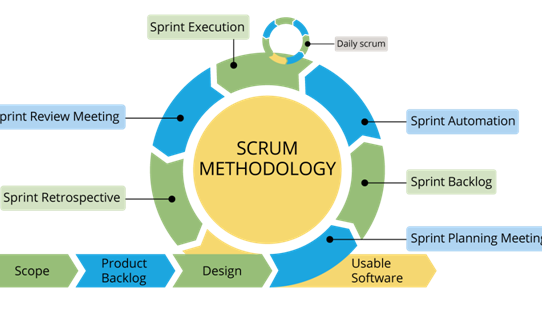
Scrum takımlarının artan performansının arkasındaki sebepler çok çeşitlidir. Birincisi, Scrum elbette olgun bir geliştirme ve proje yönetimi yapısıdır. Sık yinelemeler ve sürekli geri bildirimlerle, teslim edilen son ürünün müşteri ihtiyaçlarına uygun olduğundan emin olmanıza yardımcı olur. Gereksinimleri anında değiştirme yeteneği, projenin raydan çıkmamasını sağlar. Scrum, iyi süreçlere ek olarak psikolojik faydalarıyla da birlikte gelir. Takımların performanslarını geliştirmelerine yardımcı olmak için psikolojik mekanizmalar ve akıllı motivasyon üzerine inşa edilir.

**Scrum'ın bazı özellikleri:**

* Geliştirme süreci boyunca paydaşlardan, müşterilerden ve son kullanıcılardan toplanan geri bildirimler.
* Değişen gereksinimlere hızlı tepki.
* 1-4 haftalık sprintler, geliştirme ekibinin hızını artırarak aciliyet duygusu sağlar.
* Sevk edilebilir artımlar erken ve sıklıkla sunulur ve paydaşların geri bildirim sağlamasına olanak tanır.
* Üretken, etkili ve verimli toplantılar (günlük Scrum toplantıları ve retrospektifler) geliştiricilerin ihtiyaç duyduklarında yardım almalarını sağlamaya yardımcı olur.
* Scrum, çoğumuzun paylaştığı temel bir özellikten tam olarak yararlanır: son teslim tarihlerini kaçırmak istemiyoruz. Temel insan psikolojisi üzerine inşa edilen Scrum, bir dizi "daha küçük" son teslim tarihi oluşturmak için işi kısa yinelemelere böler ve böylece sürekli bir aciliyet duygusu oluşturur.

Ancak sadece bu da değil: Yüz yüze toplantıların (diğer geliştiriciler, müşteriler ve hatta son kullanıcılarla) yoğun kullanımı sayesinde, geliştiriciler işlerinin ne kadar önemli olduğuna dair gerçek bir fikir edinirler. Ekip üyeleri, ilerlemelerini rapor etmeli, yapılacakları açıklamalı ve günlük standuplar sırasında karşılaştıkları sorunları açıklamalıdır. Geriye dönük toplantılar, yapısal sorunların çözülmesi için süreçlerin sürekli iyileştirilmesine yardımcı olur. Sorumluluk duygusu, Scrum'ın yönetim görevlerini takıma devretme uygulamasıyla daha da derinleşir. Sorumluluk, elbette, geliştiriciler için hem bir yük hem de bir nimettir. Böylesine hızlı tempolu bir ortamda üretkenliğe odaklanmak tükenmişliğe yol açabilir, ancak kendi kendini yönetme ve yönetim süreçlerine daha fazla katılım ve dürüst ve açık iletişimi teşvik eden günlük toplantılar ile Scrum takımları, parmakları işaret etmektense olumlu pekiştirmeye ve engelleri kaldırmaya odaklanan destekleyici iklim oluştururlar

Sonuç olarak, Scrum basitliği ve yüksek performansı nedeniyle en popüler Çevik proje yönetimi yapısıdır. Bir ekip çalışması ortamında yapılan bir başarı duygusu, olumlu geri bildirim ve yapılan işin sahiplenilmesi ihtiyacından yararlanır.



Kaynakça

<https://medium.com/@omerharuncetin/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BC-modelleri-543c7879a742>

<https://phoenixnap.com/blog/software-development-life-cycle#:~:text=Software%20Development%20Life%20Cycle%20is,%2C%20Test%2C%20Deploy%2C%20Maintain>.

<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://medium.com/@melsatar/software-development-life-cycle-models-and-methodologies-297cfe616a3a>

<https://www.scrum.org/resources/what-is-scrum>

<https://caglartelef.com/yazilim-yasam-dongusu/>

<https://www.guru99.com/software-engineering-prototyping-model.html#:~:text=Prototyping%20Model%20is%20a%20software,are%20not%20known%20in%20detail>.

<https://tr.wikipedia.org/wiki/Scrum>

Medium linki: <https://hazrkatircioglu.medium.com/yaz%C4%B1l%C4%B1m-ya%C5%9Fam-d%C3%B6ng%C3%BCs%C3%BC-modelleri-software-development-life-cycle-models-2a5358567a5>