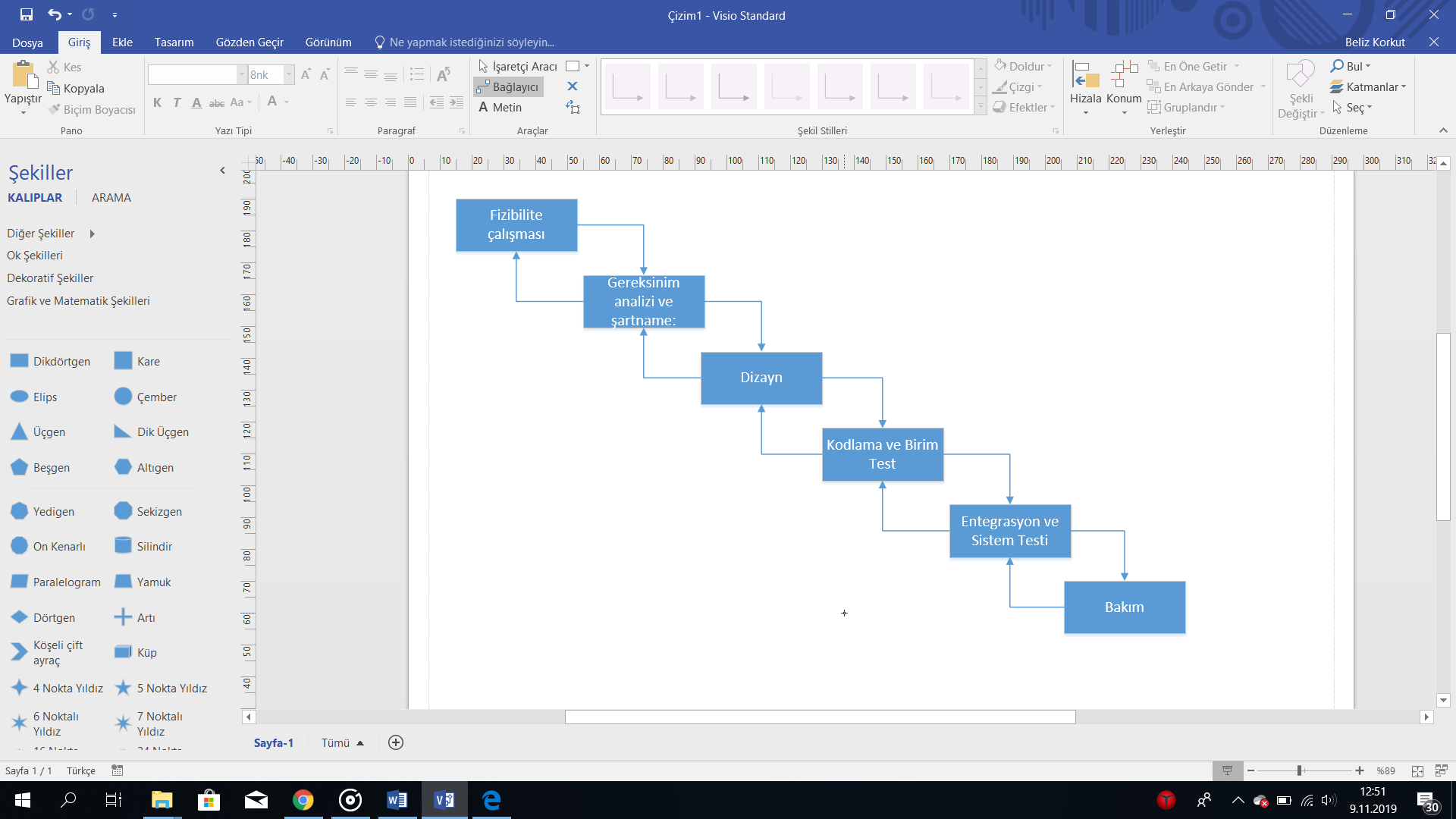
# Projemiz İçin Yazılım Geliştirme Medolojilerinin Kıyaslanması:

## Şelale Modeli:

Temel yazılım geliştirme yaşam döngüsü modelidir. Eskiden bu model çok popülerdi ama günümüzde az kullanılmaktadır. Ancak diğer tüm yazılım geliştirme modellerine temel olmuştur.

Klasik şelale modeli yaşam döngüsünü bir evreye böler. Bu model önceki aşama tamamlandıktan sonra bir aşama başlatılabileceğini düşünmektedir. Bu, bir aşamanın çıktısının bir sonraki aşamaya gelişi olacaktır. Böylece gelişme süreci şelale içinde sıralı bir akış olarak kabul edilebilir. Buradaki aşamalar birbirleriyle örtüşmez.

Şelale Modelinin yazılım geliştirme sürecini doğrusal sıralı bir akışta gösterdiği gibi; bu nedenle aynı zamanda Doğrusal Sıralı Yaşam Döngüsü Modeli olarak da adlandırılır.



* **Fizibilite Çalışması**: Bu aşamanın temel amacı, yazılımı geliştirmenin finansal ve teknik olarak mümkün olup olmayacağının belirlenmesidir. Fizibilite çalışması, sorunu anlamayı ve sonra sorunu çözmek için çeşitli olası stratejileri belirlemeyi içerir. Tanımlanan bu farklı çözümler, faydaları ve zararları temelinde analiz edilir, En iyi çözüm seçilir ve diğer tüm aşamalar bu çözüm stratejisine göre gerçekleştirilir.
* **Gereksinim analizi ve şartname**: Gereksinim analizi ve şartname aşamasının amacı, müşterinin kesin ihtiyaçlarını anlamak ve bunları doğru bir şekilde belgelendirmektir. Bu aşama iki farklı aktiviteden oluşur.
  + **İhtiyaç toplama ve analiz**: Önce yazılıma ilişkin tüm gereksinimler müşteriden toplanır ve sonra toplanan gereksinimler analiz edilir. Amacı, eksikleri ve tutarsızlıkları ortadan kaldırmaktır.
  + **Gereksinim özellikleri**: Bu analiz edilen gereksinimler bir yazılım gereksinimi belirtimi (SRS) belgesinde belgelenmiştir. SRS belgesi, geliştirme ekibi ve müşteriler arasında bir sözleşmedir. Müşteriler ve geliştiriciler arasındaki gelecekteki herhangi bir anlaşmazlık SRS belgesi incelenerek çözülebilir.
* **Dizayn:** Amacı, SRS belgesinde belirtilen gereksinimleri, bazı programlama dillerinde uygulamaya uygun bir yapıya dönüştürmektir.
* **Kodlama ve Birim Test:** Kodlama aşamasında yazılım tasarımı, uygun herhangi bir programlama dili kullanılarak kaynak koduna çevrilir. Böylece tasarlanan her modül kodlanır. Birim test aşamasının amacı, her bir modülün düzgün çalışıp çalışmadığını kontrol etmektir.
* **Entegrasyon ve Sistem Testi:** Uygulama aşamasında geliştirilen tüm birimler her birimin test edilmesinden sonra bir sisteme entegre edilmiştir. Tasarlanan yazılımın, herhangi bir kusur veya hata olup olmadığını bulmak için sürekli yazılım testinden geçmesi gerekir. Yazılım yüklenirken müşterinin herhangi bir sorunla karşılaşmaması için testler yapılır. Sistem testi üç farklı test faaliyetinden oluşur:
  + **Alfa testi:** Alfa testi, geliştirme ekibi tarafından gerçekleştirilen sistem testidir.
  + **Beta testi:** Beta testi, müşteriler tarafından yapılan sistem testidir.
  + **Kabul testi:** Yazılım teslim edildikten sonra müşteri, teslim edilen yazılımı kabul edip etmemeyi veya reddetmeyi belirlemek için kabul testini gerçekleştirir.
* **Bakım:** bir yazılım yaşam döngüsünün en önemli aşamasıdır. Bakım için harcanan çaba, tam bir yazılım geliştirmek için harcanan toplam çabanın% 60'ıdır. Temel olarak üç tür bakım vardır:
  + **Düzeltici Bakım**: Bu bakım türü, ürün geliştirme aşamasında bulunamayan hataları düzeltmek için yapılır.
  + **Mükemmel Bakım**: Bu tür bir bakım, müşterinin isteğine göre sistemin işlevselliğini artırmak için yapılır.
  + **Uyarlanabilir Bakım**: Yazılımı, yeni bir ortamda (yeni bir bilgisayar platformunda veya yeni bir işletim sisteminde) çalışmak için gereklidir.

|  |  |
| --- | --- |
| Şelale Modelinin Avantajları: | Şelale Modelinin Dezavantajları: |
| Çok basittir ve de anlaşılması kolaydır. Bu nedenle diğer yazılım geliştirme yaşam döngüsü modelleri için temel olmuştur. | **Geliştirme sürecinin her aşaması için zaman ve maliyet tahmin etmek zordur.** |
| Bu modeldeki adımlar birer birer işlenir. | **Geri besleme yok: Hiçbir aşamada geliştiriciler tarafından hiçbir hata yapılmayacağını varsayılır. Bu nedenle, hata düzeltme için herhangi bir mekanizma içermez.** |
| Her aşama açıkça tanımlanmıştır. | **Değişim taleplerini yerine getirmek zor: Bu model, tüm müşteri gereksinimlerinin projenin başlangıcında tamamen ve doğru bir şekilde tanımlanabileceğini varsaymaktadır, ancak aslında müşterilerin gereksinimleri zaman içinde değişmeye devam etmektedir. Gereksinim belirtim aşaması tamamlandıktan sonra herhangi bir değişiklik talebinde bulunduğunda bunları karşılamak zordur.** |
| Süreç, eylemler ve sonuçlar çok iyi belgelenmiştir. | **Örtüşen adımlar yok: Yeni bir adımın sadece önceki adımın tamamlanmasından sonra başlayabilmesini önermektedir. Ancak gerçek projelerde bu sürdürülemez. Verimliliği artırmak ve maliyeti düşürmek için, adımlar üst üste gelebilir.** |
| Bu model daha küçük projeler ve gereksinimlerin iyi olduğu projeler için iyi çalışır. | **Karmaşık ve nesneye yönelik projeler için iyi bir model değil.**  **İhtiyaçların orta ila yüksek değişim riski altında olduğu projeler için uygun değildir.** |

## Helezonik Model (Spiral Model):

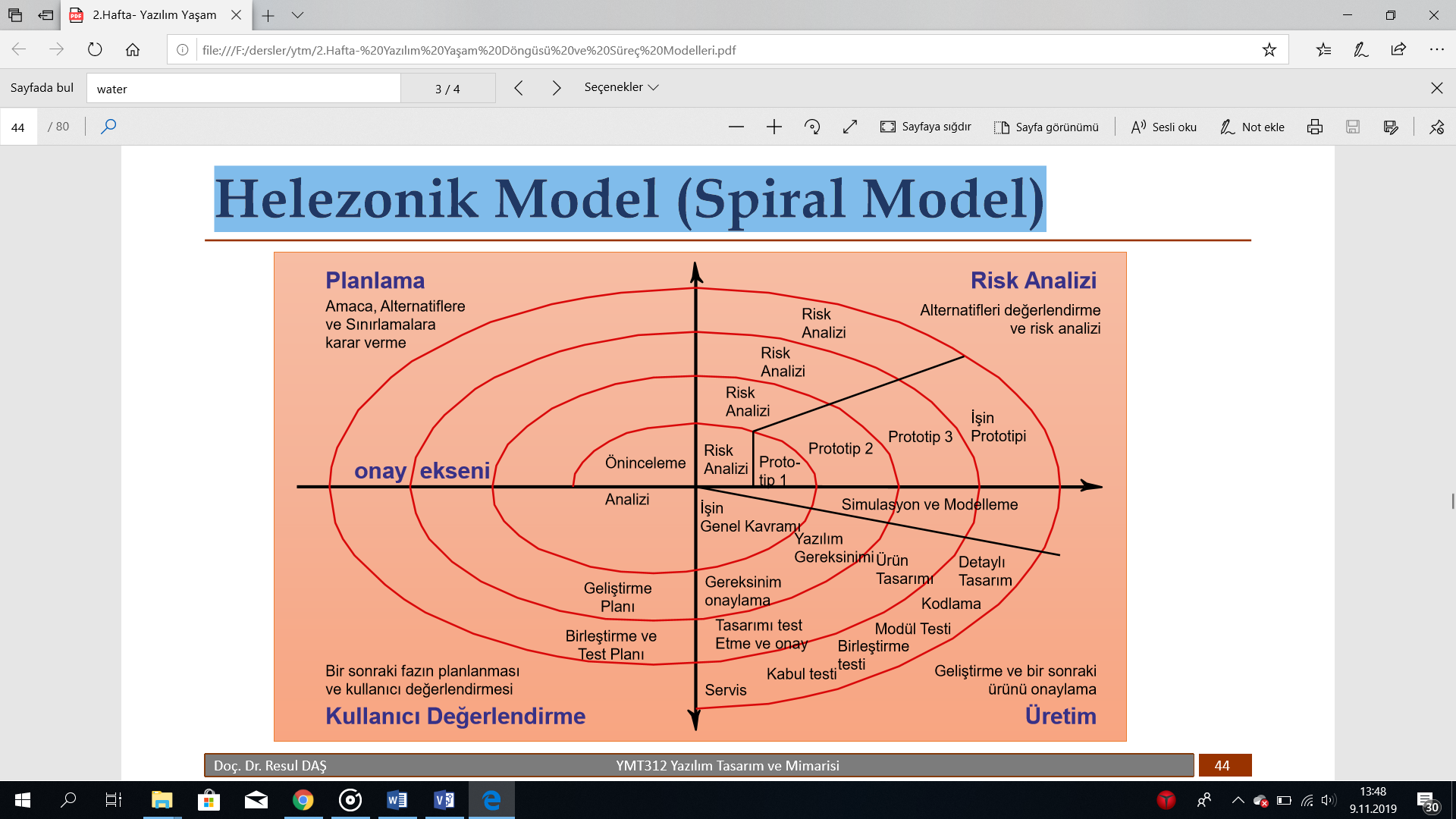
Spiral model, şelale modeli ve yinelemeli modelin bir birleşimidir. Yani şelale modelinin risk analizine çok önem veren bir birleşimidir Spiral modeldeki her aşama bir tasarım hedefi ile başlar ve müşterinin ilerlemeyi gözden geçirmesiyle sona erer.

|  |  |
| --- | --- |
| Şelale ve Spiral Model Arasındaki Farklar: | |
| Şelale modeli, yazılım projeleri geliştirmek için nispeten doğrusal bir sıralı tasarım yaklaşımıdır. | Spiral model, yazılım projeleri için risk odaklı bir süreç modeli üretecidir. |
| Şelale modelinde hatalar veya riskler tanımlanır ve aşamalar tamamlandıktan sonra giderilir. | Spiral modelde hatalar veya riskler önceden tanımlanır ve giderilir. |
| Şelale modelinde müşteri katılımı minimumdur. | Spiral modelde, müşteri katılımı yüksektir. Müşteri, ürünün ne olduğu konusunda farkındalığa sahiptir. |
| Şelale modelinde, bir evreyi tamamladıktan ve yeni bir evreye ulaştıktan sonra, bir önceki evreye geri dönmek mümkün değildir. | Spiral model yinelemelerle çalışır, bu yüzden önceki aşamalara geri dönmek mümkündür. |
| Şelale modeli küçük projeler için ve net gereksinimleri olan projeler için kullanılabilir. | Spiral model, sürekli risk analizi gerektiren büyük, karmaşık projeler için kullanılabilir. |
| Şelale modeli basit ve kolaydır. | Spiral model karmaşık bir modeldir. |

Diyagram temsili ile birçok spiral şeklinde görünür. Spiralin düğüm sayısı bilinmemektedir ve projeden projeye değişebilir. Spiralin her halkasına yazılım geliştirme sürecinin bir aşaması denir. Ürünü geliştirmek için gereken adım sayısı, proje risklerine bağlı olarak proje yöneticisi tarafından değiştirilebilir.

Modelindeki geliştirme ekibi, küçük bir gereksinim ile başlar ve bu gereksinimler kümesi için her geliştirme aşamasından geçer. Yazılım mühendisliği ekibi, uygulama üretim aşamasına hazır olana kadar her artan spiralde ek gereksinim için işlevsellik eklenebilir.

Spiralin yarıçapı herhangi bir noktada projenin giderlerini (maliyetini) ve şimdiye kadar kaydedilen ilerlemeyi temsil eder.



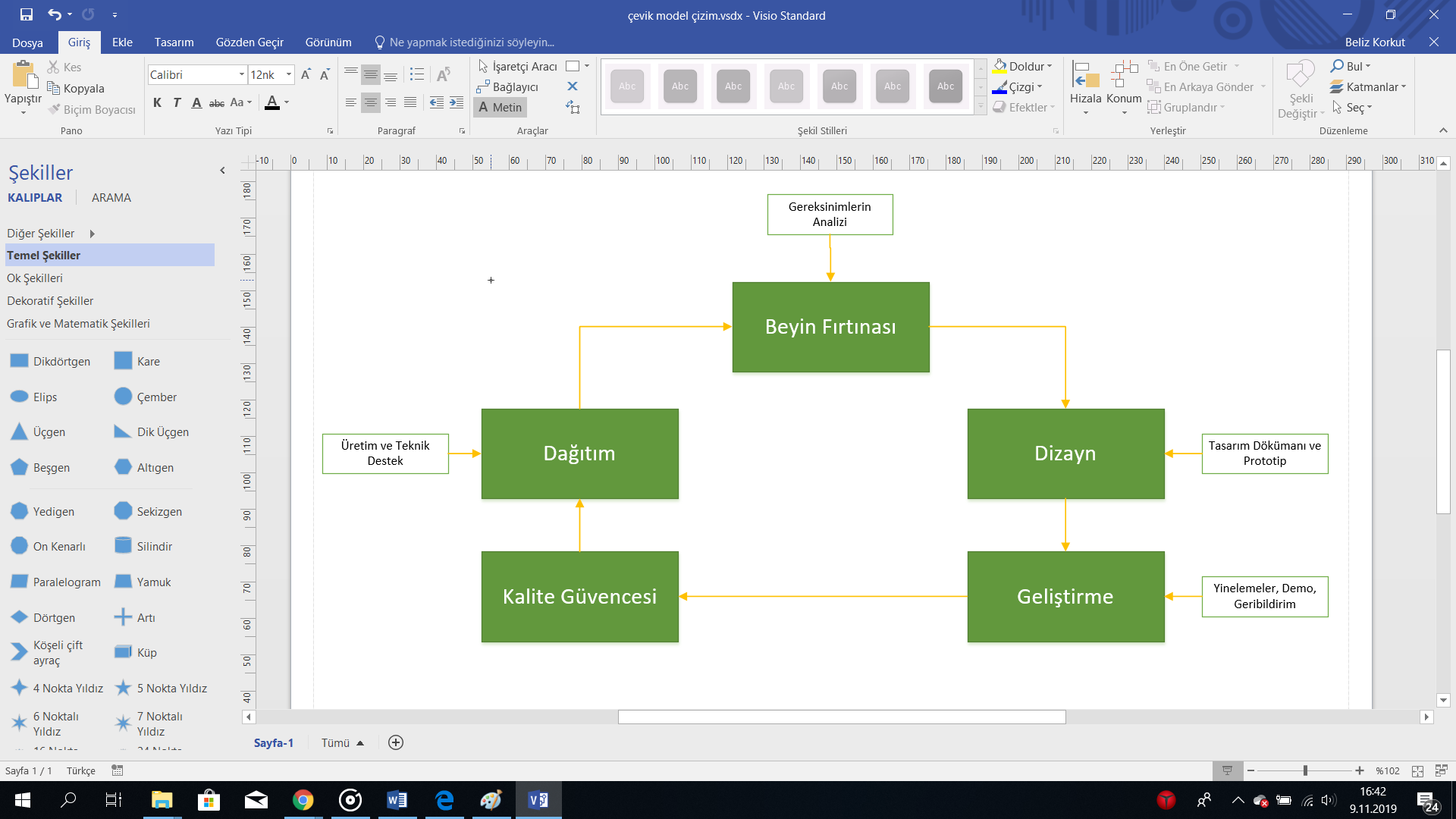
* **Planlama:** Hedefler ve alternatif çözümlerin belirlenir. Müşterilerden gereksinimler toplanır ve hedefler her aşamada tanımlanıp geliştirilir ve analiz edilir. Yinelemenin maliyetini, zamanlamasını ve kaynaklarını tahmin etmeyi de içerir.
* **Risk Analizi:** Bu süreç boyunca mümkün olan en iyi çözümü seçmek için tüm olası çözümler değerlendirilir. Ardından bu çözümle ilişkili riskler tanımlanır ve riskler mümkün olan en iyi strateji kullanılarak çözülür. Bu evrenin sonunda prototip, mümkün olan en iyi çözüm için inşa edilmiştir.
* **Üretim:** Müşteri yerinde test etme, kodlama ve yazılım dağıtımı içerir.
* **Kullanıcı Değerlendirmesi:** Müşteriler yazılımın şimdiye kadar geliştirilen versiyonunu değerlendiriyor. Sonunda bir sonraki aşama için planlama başlatılır. Ayrıca, zamanlama kayması ve maliyet aşımı gibi risklerin tanımlanmasını ve izlenmesini de içerir.

|  |  |
| --- | --- |
| Spiral Modelinin Avantajları: | Spiral Modelinin Dezavantajları: |
| Büyük ve karmaşık ve riskli projelerde kullanılması önerilir. | **Kullanılması pahalı bir model olabilir.** |
| Risk Analizi: Risk, bir yazılım projesinin başarıyla tamamlanmasını etkileyebilecek her türlü olumsuz durumdur. Spiral modelin en önemli özelliği, proje başladıktan sonra bu bilinmeyen risklerin üstesinden gelmektir. Spiral model, yazılım geliştirmenin her aşamasında bir prototip oluşturma kapsamı sağlayarak risklerle başa çıkmayı destekler. | **Projenin başarısı, risk analizi aşamasına büyük ölçüde bağlıdır bu yüzden risk analizi oldukça özel bir uzmanlık gerektirir.** |
| Gereksinimde Esneklik: Daha sonraki aşamalarda Gereksinimlerdeki değişiklik istekleri bu model kullanılarak doğru bir şekilde dahil edilebilir. | **Proje başlangıcında adım sayısı bilinmediğinden, zaman tahmini çok zordur. Ve de spiral süresiz olarak devam edebilir.** |
| Müşteri, yazılım geliştirme sürecinin ilk evresinde ürünün gelişimini görebilir ve böylece kullanıcılar ürünün tamamlanmasından önce onu kullanarak sisteme alışmış olur. | **Spiral Model, diğer süreç modellerinden çok daha karmaşıktır. Spiral Model pahalı olduğu için küçük projeler için uygun değildir.** |

## Genel Olarak Çevik Yazılım Modelli:

Çevik model, öncelikli olarak değişen taleplere hızla uyum sağlamak için bir projeye yardımcı olmak için tasarlanmıştır. Dolayısıyla, Çevik modelin asıl amacı, hızlı proje tamamlamayı kolaylaştırmaktır. Görevler daha küçük yinelemelere bölünüp veya parçalar doğrudan uzun süreli planlamayı içermez. Bu da sürecin projeye uyması, belirli bir proje için gerekli olmayabilecek faaliyetlerin kaldırılmasıyla sağlanır. Ayrıca, zaman ve emek kaybı olan her şeyden kaçınılır. Çevik modelde, gereksinimler aşamalı olarak geliştirilebilecek birçok küçük parçaya ayrıştırılır. Çevik model, yinelemeli gelişimi benimser. Her artımlı kısım, bir yineleme üzerinde geliştirilir. Her yinelemenin küçük ve kolay yönetilebilir olması ve sadece birkaç hafta içinde tamamlanabilmesi amaçlanmıştır. Bir yineleme planlanır, geliştirilir ve müşterilere dağıtılır. Uzun vadeli planlar yapılmamıştır.

Çevik model, yinelemeli ve artımlı işlem modellerinin birleşimidir. Çevik süreç modelinde kısa süreli bir "çerçeve" olarak kabul edilir. Tüm projenin daha küçük parçalara bölünmesi, proje riskini en aza indirmeye ve toplam proje teslim süresi gereksinimlerini azaltmaya yardımcı olur. Her yineleme, çalışan bir müşteriye müşteriye gösterilmeden önce planlama, ihtiyaç analizi, tasarım, kodlama ve test dâhil tam bir yazılım geliştirme yaşam döngüsü boyunca çalışan bir ekibi içerir.



* **Gereksinimlerin toplanması**: Bu aşamada gereksinimleri tanımlanır. İş fırsatlarını açıklamalı ve projeyi oluşturmak için gereken zamanı ve çaba planlanır. Bu bilgilere dayanarak, teknik ve ekonomik fizibilite değerlendirmesi yapılır.
* **Gereksinimleri tasarlanması:** Projeyi tanımlandıktan sonra gereksinimleri tanımlamak için müşteri ile birlikte çalışılır. Yeni özelliklerin çalışmasını göstermek ve mevcut sisteminize nasıl uygulanacağını göstermek için kullanıcı akış şemasını veya üst seviye UML şemasını kullanılabilir.
* **Yapım / yineleme**: Takım gereksinimleri tanımladığında çalışma başlar. Tasarımcılar ve geliştiriciler, çalışan bir ürün dağıtmayı amaçlayan projeleri üzerinde çalışmaya başlar. Ürün çeşitli iyileştirme aşamalarından geçecek, bu nedenle basit, minimum işlevsellik içerir.
* **Test Etme:** Bu aşamada, Kalite Güvence ekibi ürünün performansını inceler ve hatayı arar.
* **Dağıtım:** Bu aşamada, ekip, kullanıcının çalışma ortamı için bir ürün yayınlar.
* **Geri bildirim**: Ürünü piyasaya sürdükten sonra son adım geri bildirimdir. Bu ekipte ürün hakkında geri bildirim alır ve geri bildirim yoluyla çalışır.

**Çevik modelin ilkeleri:**

* Geliştirme sırasında müşteri ile yakın temas kurmak ve çeşitli gereksinimlerin açık bir şekilde anlaşılmasını sağlamak için, her çevik proje genellikle ekipte bir müşteri temsilcisi içerir. Her yineleme müşterinin incelemesinin sonunda, ilerleme kaydedilir ve gereksinimleri yeniden değerlendirilir.
* Çevik model, kapsamlı dokümantasyon yerine çalışan yazılım dağıtımına dayanır.
* Yazılımın artan sürümlerinin birkaç hafta arayla müşteri temsilcisine sık teslimatı olur.
* Müşteriden gelen talep değişikliği talepleri teşvik edilir ve verimli bir şekilde dahil edilir.
* Etkin ekip üyelerine sahip olmak ve aralarındaki iletişimi geliştirmek üzerinde daha fazla önem verildiğini vurgulamaktadır. Geliştirme ekibi üyeleri arasında gelişmiş iletişimin, resmi belgelerin değişimi yerine yüz yüze iletişim yoluyla sağlanabileceği anlaşılmıştır.
* Ekip üyelerinin yüz yüze iletişimde anlamlı bir şekilde yer almasına ve işbirliğine dayalı çalışma ortamına sahip olmalarına yardımcı olmak için geliştirme ekibi boyutunun küçük tutulması (5 - 9 kişi) önerilir.
* Çevik geliştirme süreci genellikle Çift Programlama'yı kullanır. Çift programlamada, iki programcı bir iş istasyonunda birlikte çalışır. Biri kod yazarken, diğeri yazıldığı gibi kodu gözden geçirir. İki programcı rollerini saat başı veya ragele değiştirir.

### Çevik Yazılım Metotlarından Scrum Modeli:

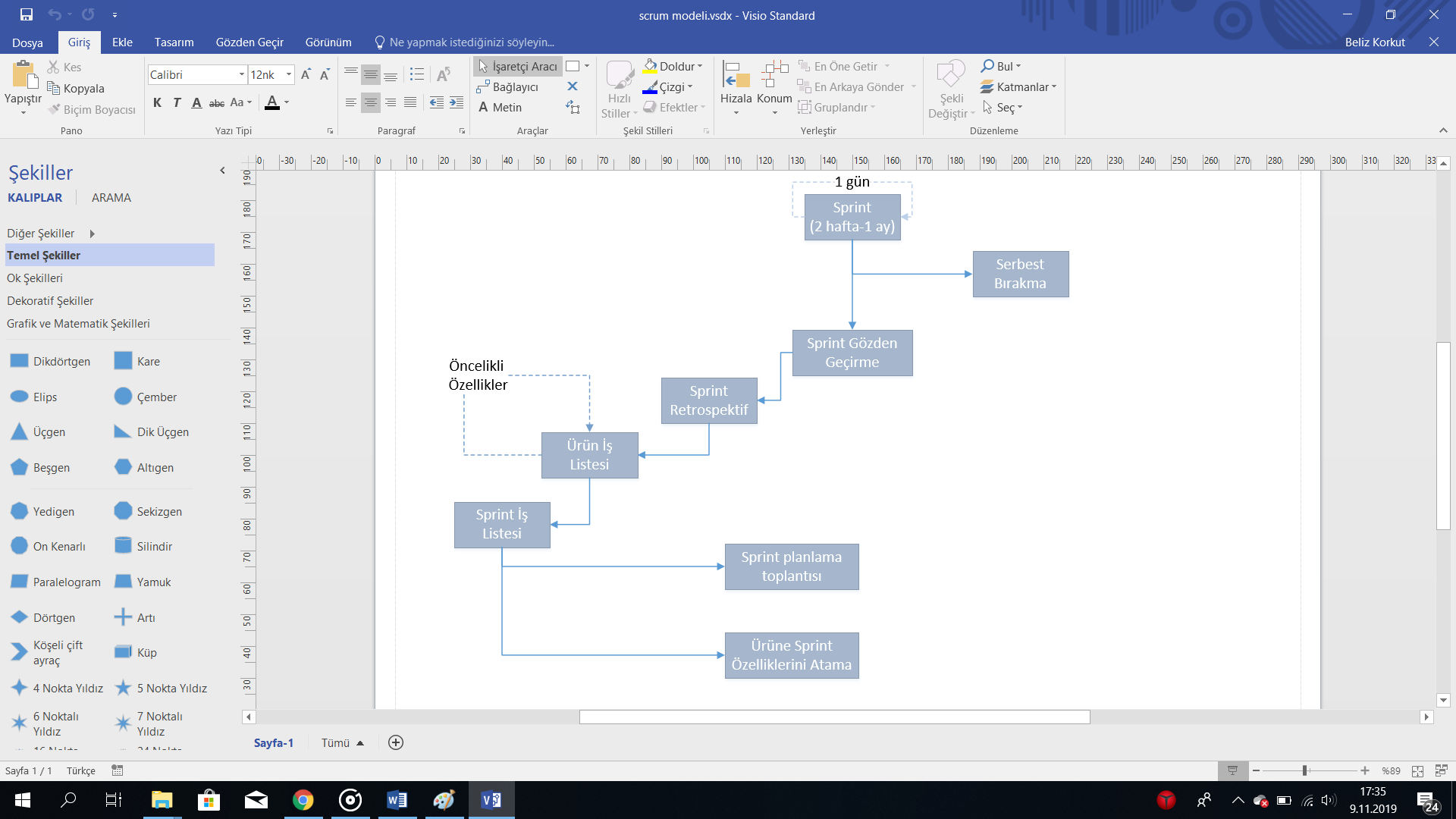
Scrum üç rolden oluşur:

* Ürün Sahibi: Ürün sahibi, şirketi temsil eden bireydir. Scrum yazılım geliştirmesinde işletmeyi, müşterileri veya kullanıcıları temsil eder ve ekibi doğru ürünü oluşturmaya yönlendirir.
* Scrum Ustası: Scrum ustası, süreci kontrol etmek ve ölçmekle görevli proje yöneticisidir.
* Ekip: Ekip, proje görevlerini yerine getirecek olanlardan oluşur.

Bir projede her şeyin nasıl yapılacağına dair eksiksiz ve ayrıntılı açıklamalar sağlamak yerine, çoğu Scrum yazılım geliştirme ekibine bırakılmıştır. Bunun nedeni, ekibin sundukları problemi nasıl çözeceğini en iyi şekilde bilmesidir. Scrum, kendini organize eden, işlevsel bir takıma dayanır. Scrum takımı kendi kendini organize ediyor, çünkü hangi kişinin hangi görevi yapacağına veya bir sorunun nasıl çözüleceğine karar veren genel bir takım lideri bulunmuyor. Bunlar takım tarafından bir bütün olarak karar verilen konulardır. Scrum'da bir takım çapraz fonksiyoneldir, yani fikirden uygulamaya kadar herkesin bir özellik kazanması gerekir.

Diğer Sucrum kavramları:

* **Ürün biriktirme listesi:** Ürün biriktirme listesi, istenen her özelliği veya üründe yapılan değişikliği içeren öncelikli bir özellikler listesidir. Açıklığa kavuşturmak için, ürün listesi, ürün için istenen özelliklerin bir listesidir.
* **Sprint İş Listesi:** Bir Sprint'te tamamlanması gereken görevlerin bir listesidir.
* **Sprint planlama toplantısı:** Her sprintin başlangıcında, ürün sahibinin ürün biriktirme listesindeki en üst kalemleri takıma sunduğu bir sprint planlama toplantısı yapılır. Scrum ekibi gelecek sprint boyunca tamamlayabilecekleri işleri seçer sonrasında da bu iş daha sonra ürün topluluğundan bir sprint topluluğuna taşınır, bu da takımın sprint içinde tamamladığı taahhüt ettiği ürün biriktirme öğelerini tamamlamak için gereken görevlerin listesidir.
* **Günlük Scrum:** Sprint boyunca her gün, günlük scrum adı verilen kısa bir toplantı yapılır. Bu toplantı, günlük çalışmaların içeriğinin belirlenmesine yardımcı olur ve ekibin yolda kalmasına yardımcı olur. Tüm ekip üyelerinin günlük işlere katılmaları gerekir.
* **Sprint inceleme toplantısı**: Her sprintin sonunda takım, tamamlanan işlevselliği bir sprint inceleme toplantısında gösterir; bu sırada ekip, sprint boyunca neler yapıldığı gösterilir.
* **Sprint retrospektif**: Ayrıca her sprintin sonunda, takım (ScrumMaster ve ürün sahibi dahil) ekibin kendileri için ne kadar iyi çalıştıklarını ve ne gibi değişiklikler yapmak istediklerini düşündükleri bir toplantı olan sprint retrospektifini gerçekleştirir.



* **Sprint**: Sprint, bir ay veya daha kısa bir zaman kutusudur. Önceki Sprint tamamlandıktan hemen sonra yeni bir Sprint başlar.
* **Serbest bırakma**: Ürün tamamlandığında, Serbest bırakma aşamasına geçer.
* **Sprint Gözden Geçirmesi**: Ürün hala elde edilemez özelliklere sahipse, bu aşamada kontrol edilecektir.
* **Sprint Retrospektif**: Bu aşamada ürünün kalitesi veya durumu kontrol edilir.
* **Ürün İş Listesi:** Öncelikli özelliklere göre ürün organize edilmiştir.
* **Sprint İş Listesi:** Sprint planlama toplantısı için ürüne atanan özelliklerdir.

|  |  |
| --- | --- |
| Scrum Modelinin Avantajları: | Scrum Modelinin Dezavantajları: |
| Scrum zamanın ve paranın etkin kullanımını sağlar. Sprint süreci, proje tüm hızıyla devam ederken bile satılabilir bir ürünle sonuçlanabilir. | **Scrum’un kesin bir bitiş tarihi yoktur. Sadece gereksinim, her bir kişi veya ekibin beklentileri karşılaması önemlidir.** |
| Büyük projeler kolayca yönetilebilir sprintlere ayrılır. Sprint incelemesi sırasında gelişmeler kodlanmış ve test edilmiştir. | **Bireyler çok kararlı olursa ya da işbirliğinden kaçınırlarsa, proje başarısızlığı ihtimali yüksektir.** |
| Hızlı gelişen kalkınma projeleri için iyi çalışır. | **Sadece deneyimli ekip üyeleriyle başarılı olabilir.** |
| Scrum, müşterilerden geri bildirim alır. | **Herhangi bir ekip üyesi bir projenin ortasından ayrılırsa, proje üzerinde çok büyük olumsuz bir etkisi olabilir.** |
| Kısa sprintler ve sürekli geri bildirimlerle, değişikliklerle baş etmek ve uyum sağlamak daha kolaydır. Örneğin, ekip bir sprint sırasında yeni bir kullanıcı hikayesi keşfederse, bu ayrıntıyı bir sonraki sprinte kolayca ekleyebilir. | **Kötü tanımlanmış görevler yanlışlıklara yol açabilir: Görevler iyi tanımlanmadığında proje maliyetleri ve zaman çizelgeleri doğru olmaz. İlk hedefler net değilse, planlama zorlaşır ve sprintlerin başlangıçta tahmin edilenden daha fazla zaman alabilir.** |
| Sürekli iletişim, ekibin ortaya çıkar çıkmaz tüm sorunların ve değişikliklerin farkında olmasını sağlar, böylece masrafları azaltmaya ve kaliteyi yükseltmeye yardımcı olur. Küçük parçalardaki özellikleri kodlayarak ve test ederek sürekli geri bildirim vardır ve düzeltmek için çok pahalılaşmadan önce hatalar erkenden düzeltilebilir. | **Scrum Master'ın yönettikleri takıma güvenmeleri ve ne yapacaklarını asla söylememeleri gerekir. Eğer Scrum Master ekibi kontrol etmeye çalışırsa, proje başarısız olabilir.** |
| Scrum Takımı'na ne zaman ve ne yapmaları gerektiğini söyleyen hiçbir proje yöneticisi yok. Bunun yerine, ekip topluca her sprint içinde ne iş başarabileceğine karar verir. Hepsi birlikte çalışır ve birbirlerine yardım eder, işbirliğini geliştirir ve her ekip üyesine bağımsız olmalarını sağlar. Her takım üyesinin bireysel çabası günlük toplantılar sırasında görülebilir. | **Sık sık yapılan değişiklikler, sık ürün teslimi ve bitmiş ürünün kesin doğasına ilişkin belirsizlik projede yer alan herkes için oldukça yoğun bir proje yaşam döngüsü yaratmaktadır.** |

|  |  |
| --- | --- |
| Scrum Model ile Şelale Modelinin Kıyaslaması: | |
| Proje geliştirme yaşam döngüsünü sprintlere ayırır. | Yazılım geliştirme süreci farklı aşamalara ayrılmıştır. |
| Artımlı bir yaklaşımı izler. | Şelale metodolojisi sıralı bir tasarım sürecidir. |
| Ürün Sahibi, Scrum Yöneticisi, Çapraz fonksiyonlu ekiplerden oluşur. | Analist, teslimat yöneticisi, proje müdürü, geliştirici, test ekibi, tasarımcılardan oluşur. |
| Test planı her sprintten sonra gözden geçirilir. Yazılım geliştirme ile eş zamanlı olarak test yapılır. | Test planı, test aşamasında nadiren tartışılmaktadır. Test aşaması, gerçekleştirim aşamasından sonra gelir. |
| Koordinasyonu ve işbirliğine sahip 2 ila 8 ekip üyesi arasında daha küçük ekipleri olması tercih edilir. | Eldeki daha fazla odaklanır. Takım büyüklüğü 2 ila 100’den fazla üyeye kadar değişebilir. Takım koordinasyonu çok sınırlıdır. |
| Müşteri katılımının yüksek olduğu daha işbirlikçi bir yaklaşım. | Müşteri katılımı, daha çok ilk aşamasında olur. |
| Değişime adapte olur. Daha önceden küçük sürüm zaman çizelgelerinin kapsamı tanımlandığı için değişiklik maliyeti düşüktür. | Proje geliştirme başladıktan sonra gereksinimlerin değiştirilmesi için bir alan yoktur. |
| Ekipten olan ürün sahibi, bir proje sırasında hemen hemen her gün gereksinimlerini hazırlar. | İş analizsti, projenin başlangıcından önce gereksinimleri hazırlar. |
| İlk önce en değerli özelliklerin uygulanmasına odaklanır. Uygulamada kullanım dışı özelliklere sahip olma riskini azaltır. | Gereksinimde verilen her şeyi oluşturmaya vurgu yapar. Uygulamaya büyük katkısı olmayan özelliklere sahip olma riskini artırır. |
| Sabit olmayan finansman projeleriyle iyi çalışır. Doğru ekiple, ROI (yazılım getirisi) daha kısa sürede görülebilir. | Kapsam ve anlaşma sapmalarının sınırlı olduğu sabit fiyatlı projelerle iyi çalışır. Projenin başarılı bir şekilde dağıtılmasından sonra ROI (yazılım getirisi) elde edilir. |
| Her 2-4 haftada bir Zaman Çizelgeleri yayınlanır. | Genel proje süresine bağlı olarak zaman çizelgelerini yayınlanır. |
| Gelişme, gereksinimlerin değişmesi ve gelişmesi beklenen projeler için idealdir. | Yöntemi ve gereksinimleri kesin olan ve hiç beklenmeyen değişikliklere sahip projeler için idealdir. |
| Yazılım ürününün son müşterilerinin ihtiyaçlarını karşıladığı ve müşterinin taleplerine göre kendini değiştirdiği bir ürün zihniyetini tanıtıyor. | Odağını tamamıyla projenin gerçekleştirilmesidir. |
| Test ekibi, problemler gerektirmeden değişim şartlarında yer alabilir. | Testin şartlarda herhangi bir değişiklik başlatması zordur. |

|  |  |
| --- | --- |
| Scrum Model ile Spiral Modelinin Kıyaslaması: | |
| Temel prensibi zaman ve çaba harcayan gereksiz faaliyetleri kaldırarak çevikliği sağlamaktır. | **Temel prensibi risk yönetimidir.** |
| Her bir aşamadan sonra müşteriye bir artış sağlamaya odaklanır, bu nedenle müşteri etkileşimi daha sık görülür. | **Temel olarak çeşitli beklenmeyen riskler ile ilgilidir, ancak müşteri etkileşimi daha azdır.** |
| Her bir yinelemede artımlı olarak kolayca geliştirilebilecek küçük parçalara bölünmesi kolay olan büyük projeler için uygundur. | **Projenin başlangıcında öngörülmesi zor olan çeşitli risk türlerine eğilimli projeler için uygundur.** |
| Dokümantasyona dayanmamaktadır. | **Dokümantasyon önemlidir.** |

## Projemiz İçin Bu Modeller Kullanılırsa:

## Şelale Modeli Kullanılırsa:

* Bu modeli kullanmak için öncesinde ekibimizin önceden bilgiye veya eğitime ihtiyacı yoktur.
* Her aşama için dokümantasyon olması her yazılan kodun ve testlerin arkasındaki mantığın daha iyi anlaşılmasını sağıp projenin ilerleyen aşamalarında rahat bir şekilde gözden geçirmemize olanak tanır.
* Projemizde ilerledikçe değişen ihtiyaç ve gereksinimler oluşursa bunları yerine getirmek oldukça zorlaşır. Bir aşamayı tamamladıktan sonra geri dönülemez. Test aşamasına gelindiğinde bu sorunu çözmek maliyetli olacaktır.

## Spiral Modeli Kullanılırsa:

* Risk değerlendirme uzmanlığı gereklidir. Takımın bu yönde eğitilmesi bilgilendirilmesi gerekir. Projemizin başarılı olması bu risk analizinin iyi bir şekilde tecrübe edilmesine bağlıdır.
* Projemizde değişen ihtiyaç ve gereksinimler oluşursa bunları yerine getirmek rahattır. Böylece daha rahat şekilde yeni özellikler eklenebilir.
* Prototip yapımı küçük parçalar halinde yapıldığı için daha rahat şekilde projemizin maliyetini tahmin edebiliriz.
* Müşteriden sık sık geri bildirimler alınacağından dolayı müşterinin istekleri doğrultusunda proje daha rahat şekillendirilecektir.

## Scrum Modeli Kullanılırsa:

* Sürekli olarak müşteriyle iletişim halinde olunduğu için müşteriden geç gelen istekler daha rahat analiz projeye uyarlanabilir.
* Proje bitmese bile bir sprint sonucunda belli bir süre idare edecek ürün oluşturulup sunulabilir.
* Bu modelde ekipler kategorilere göre görevlere ayrılmadığı için her elemanın her konuda bilgili olması, gerekirse her bir elemanın birbirinin eksikliğini tamamlaması gerektiğinden oldukça deneyimli proje elemanları gerekmektedir. Bu konuda projemizde kullanmamız sıkıntılara neden olabilir. O yüzden bu model ile geliştirilecekse proje elemanlarının bu konularda eğitilmesi şarttır. Projemizi yapma süresi kısıtlı olduğu için bunu yapmakta zor olur.
* Proje grubumuz kalabalıktır bu model de maksimum sekiz kişilik ekipler için uygundur.

Üçünün kıyaslanması sonucunda proje için en uygunu spiral modelinin kullanılmasıdır.