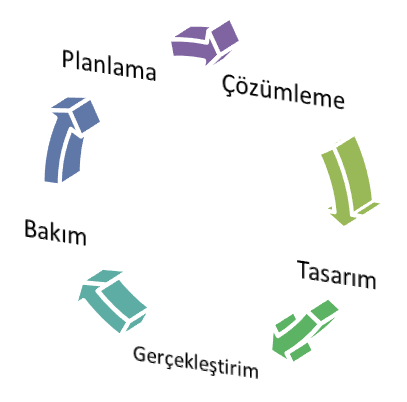
**YAZILIM YAŞAM DÖNGÜLERİ**

# Temel Yazılım Yaşam Döngüsü

Yazılım yaşam döngüsü (“Software Development Life Cycle”, SDLC), yazılım süreci boyunca gerek geliştirme ve gerek kullanım aşamasında karşımıza çıkan aşamalardır. Yazılım yaşam döngüsü, yazılım işlevleri ve ihtiyaçları sürekli değiştiği ve geliştiği için mutlaka bir döngü biçiminde düşünülmelidir. Doğrusal olarak kesinlikle düşünülmemelidir. SDLC ’ın temel aşamalarını; *Planlama, Analiz, Tasarım, Gerçekleştirme, Kullanım için Yükleme ve Bakım* olarak sınıflandırabiliriz.

Her aşama kendi içerisinde birçok farklı disiplin içerir. Açıklamalara geçmeden önce unutulmamalıdır ki her aşama üzerine derinlemesine akademik araştırmalar ve birçok yazılmış kitap vardır.



**Planlama:** Müşteriler ile görüşmeler sonucunda istediği gereksinimlerin elde edildiği ve projenin maliyeti, risk analizi, iş dağılımları, hataların onarılması ve yeni özelliklerin nasıl ekleneceği vb. birçok konunun çalışmasıdır. Bu çalışmalar sonucunda, işlerin düzgün gitmesini umarız.

**Analiz (Çözümleme):** Bu aşamada, sözleşme üzerinde çalışılır, alan uzmanları ile çalışmalar yapılır ve sistem gereksinimleri, yazılım gereksinim dokümanı oluşturulur. Var olan sistem incelenir ve temel sorunlar ortaya çıkarılır.

**Tasarım:** Sistemdeki tüm gereksinimler tamamlandıktan sonra tasarım aşamasına geçilir. Tasarım aşamasında kod söz konusu değildir. Burada sistemin ilk ve temel hali oluşturulmaya çalışır. Bir sonraki aşamaya bütün kararlar verilmiş olarak geçilmesi beklenir. Gerçekleştirme aşamasında herhangi bir soru veya karar bırakmaz.

**Gerçekleştirme:** Bu aşamada sistemin kodlanması ve testlerinin yazılacağı aşamadır. Bu aşama birçok farklı yöntem ve disiplin içerir.

**Canlıya çıkma:** Bu kısım projeden projeye farklılık gösterse de asıl amaçlanan sistemin kullanıcılara sunulmasıdır.

**Bakım:** Canlıya alınan sistemdeki hataların giderilmesi, yeni özelliklerin eklenmesi, kullanılan kütüphanelerin güncellenmesi vb. süreçlerin yapıldığı aşamadır. Bu aşama sistemin yaşamı boyunca devam eder.

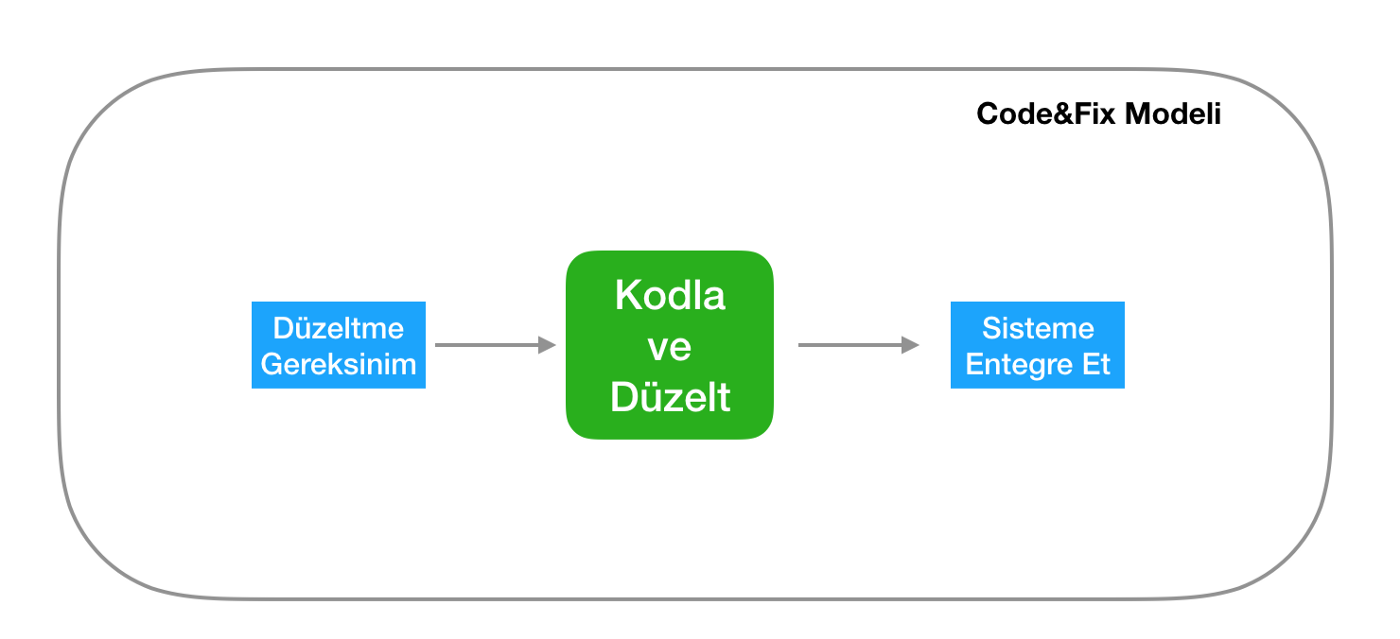
# Yazılım Yaşam Döngü Modelleri

Yazılım geliştirmenin temel döngüsü yukarıdaki gibidir. Yazılım geliştirme modelleri ise bu aşamaların sırası, tekrar edilme sayısı, ne kadar üstüne düşüldüklerine göre değişiklik gösterir, süreçlerin ilişkili ayrıntıları veya süreçler arası ilişkilerle ilgilenmezler. Üzerinde çalıştığımız projeye göre modellerde farklılık gösterir. Modellerden bahsetmek gerekirse:

* Code & Fix (Kodla ve Düzelt)
* Gelişigüzel Model
* Barok Modeli
* Waterfall (Şelale/Çağlayan) Model
* Waterfall Iterative (Şelale Yinelemli) Model
* V-Shaped (V-Şeklinde) Model
* Spiral (Helezonik) Model
* Incremental Model (Arttırımlı Model)
* Rational Unified Process
* Agile (Çevik) Model

**Code & Fix**

Problem çözmeye çalışırken yeni öğrendiğimiz çözümlerin projede uygulanıp, çıkan hataların düzeltilmesi sürecidir. En basit ürün geliştirme yöntemi (Cowboy Coding) olarak adlandırılan bu modelde en kısa sürede sonuca varma hedeflenir. Planlama, Analiz gibi kısımlar ile çok vakit harcamadan hemen ana problem üzerine odaklanılır.



## **Gelişigüzel Model**

60 lı yıllarda kullanılan Gelişigüzel Model’de, belirlenmiş bir model ya da bir yöntem bulunmaz bu yüzden model olarak adlandırmak doğru değildir. Tek kişi tarafından geliştirilen projelerde kullanılır. Projenin bakımı ve izlenebilirliği zordur.

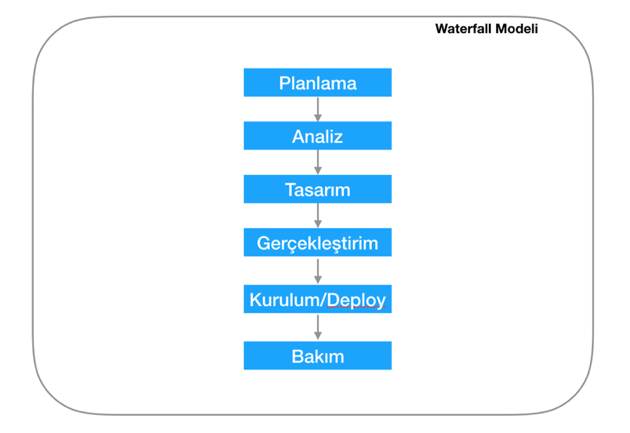
## **Barok Modeli**

70 li yıllarda kullanılan Barok modelinde, yaşam döngüsü temel adımları doğrusal olarak ele alınır ve geliştirmeler yapılır. Geri dönüşlerin nasıl olacağı tanımlı değildir. Bu modelde ayrıca dokümantasyon farklı bir süreç olarak ele alınır ve proje tamamlandıktan sonra yapılması öngörülür.

## **Waterfall (Şelale) Model**

Temel süreçlerin sırayla işletilerek bitirilme sürecidir. Geri dönülmemesi için her aşamanın doğru ve eksiksiz yapılması gerekir. En eski, en tanınmış ve en temel modeldir.

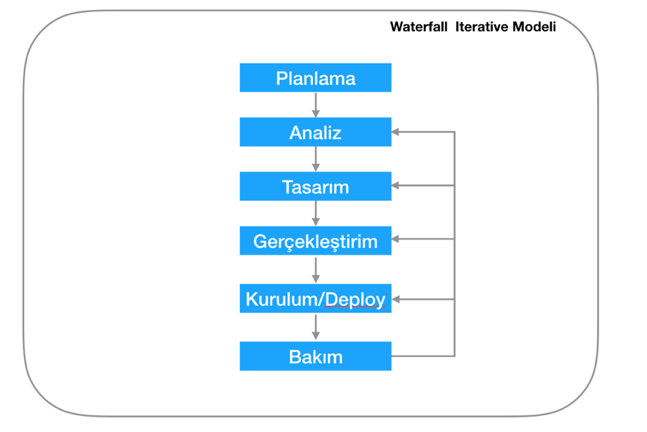
Günümüzde bu modelin işletiminin çok da basit olmadığı, analizlerin süreç içerisinde değişebildiği, kodlama sırasında tasarımın değiştiği fark edilmiş, bunun yerine daha uygulanabilir modeller aranmaya başlanmıştır.



**Waterfall Iterative (Şelale Yinelemeli) Model**

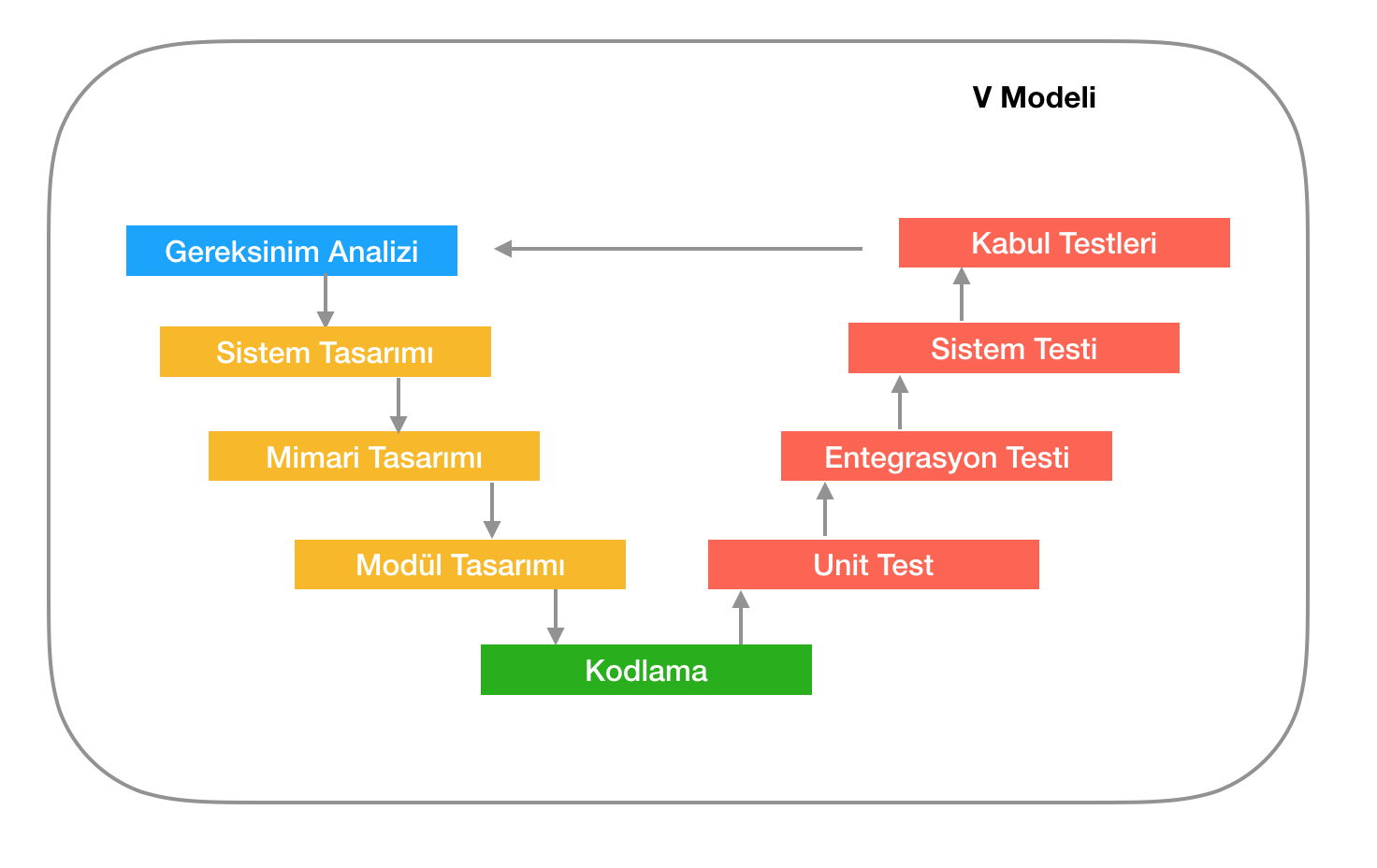
Waterfall Modelinde tüm aşamalar tamamlandıktan sonra sistemde çıkan sorunların nereden kaynaklandığını bulup, hatanın kaynaklandığı sürece dönüp tekrardan süreçlerin işletildiği modeldir. Bu modeldeki en kötü durum Analiz sürecinde kaynaklanan sorunlarda ortaya çıkacaktır. Analiz sürecine tekrardan gidilip hata düzeltilerek tüm sistem yeniden tasarlanacak, kodlanacak ve canlıya alınacaktır.

Waterfall modeli, Barok modelinden farklı olarak proje içerisinde dokümantasyonu ayrı bir süreç olarak değil de üretimin bir parçası olarak ele alır.



**V-Shaped (V-Şeklinde) Model**

Adımlar V Şeklinde oluştuğu için V-Şeklinde Model denilmiştir. Sol taraf Üretim, sağ taraf Test işlemleridir. Modelin uygulanacağı projelerde belirsizlik az, iş tanımlarının kesinlikle belirli olması gereklidir. Bu tip süreçlerde kullanıcının sistemin gelişmesinde katkısı vardır.



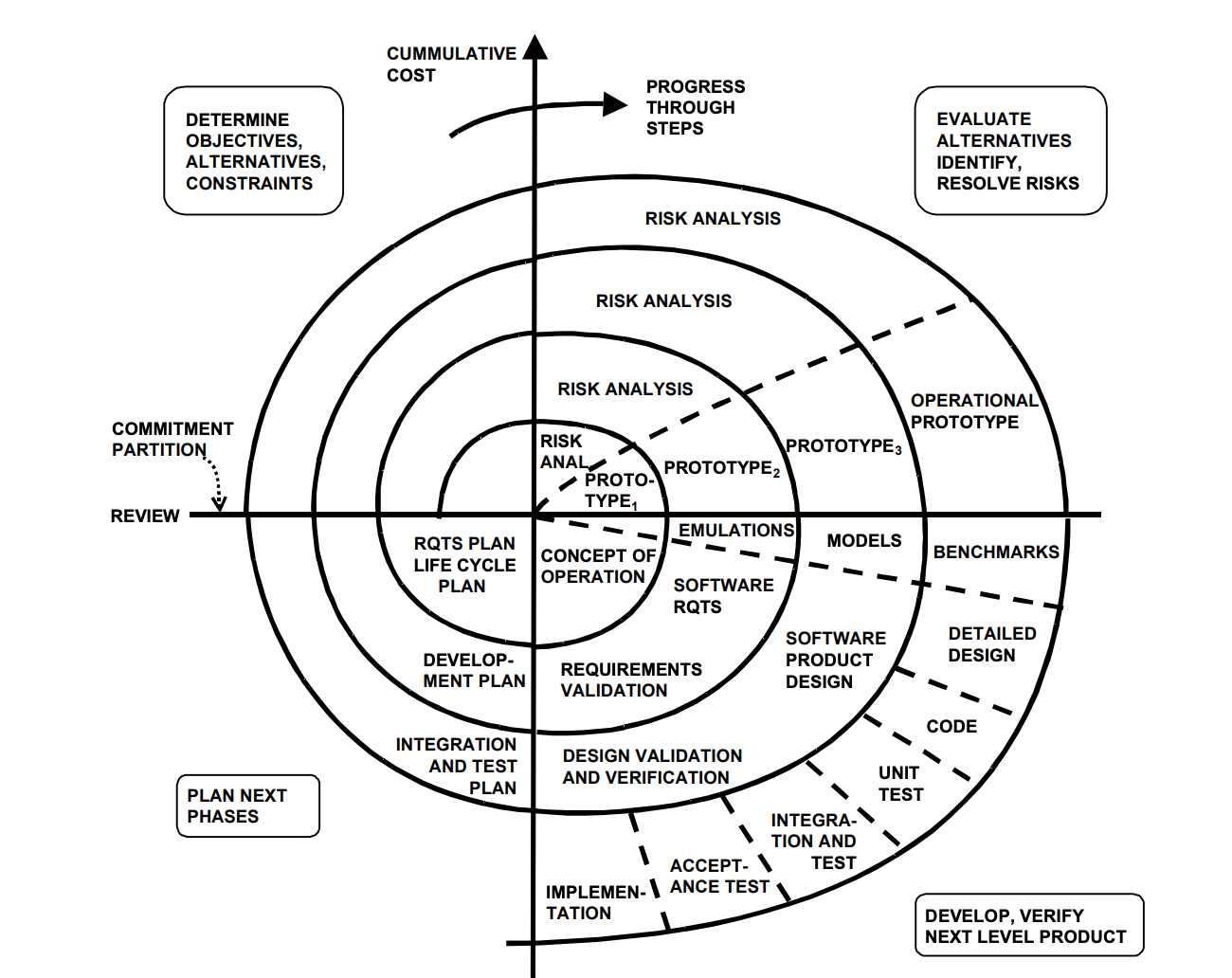
Sistemin avantajları olarak; takibi kolay, kullanımı kolay. Dezavantajları ise; aşamalarda tekrar ve risk çözümleme olmamasıdır.

**Spiral (Helezonik) Model**

Büyük projelerde planlama ve gereksinim analizini düzgün bir şekilde yapılması çok zordur çünkü karşınıza bilmediğiniz birçok risk çıkabilir. Spiral model bu riskleri en aza indirmek için diğer modellerden farklı olarak, her aşama üzerinden tekrar tekrar geçilip, her geçişte projenin ilerleme kat etmesini hedeflemektedir.

Spiral model 4 aşamadan oluşur. Bunlar;

1. Hedeflerin belirlenmesi ve Alternatif çözüm yolları üretilmesi
2. Risklerin belirlenmesi ve çözümü
3. Geliştirme ve Test
4. Sonraki Dönüşün Planlanması



Bu modelde bir döngü içerisinde sürekli dönülür. Yukarı doğru gittikçe maliyetin artması ve sola doğru gittikçe de gözden geçirme süreçlerinin artması söz konusudur.

Spiral model avantajları olarak; kullanıcının sistemi erken görmesi, hataların erken giderilmesi, parçalı geliştirme yapıldığından ilk olarak risklerin çözümlenmesi gösterilebilir. Dezavantajları ise; kompleks bir yapıya sahiptir, küçük ve az riskli projeler için pahalı bir modeldir ve fazla dokümantasyon oluşturur.

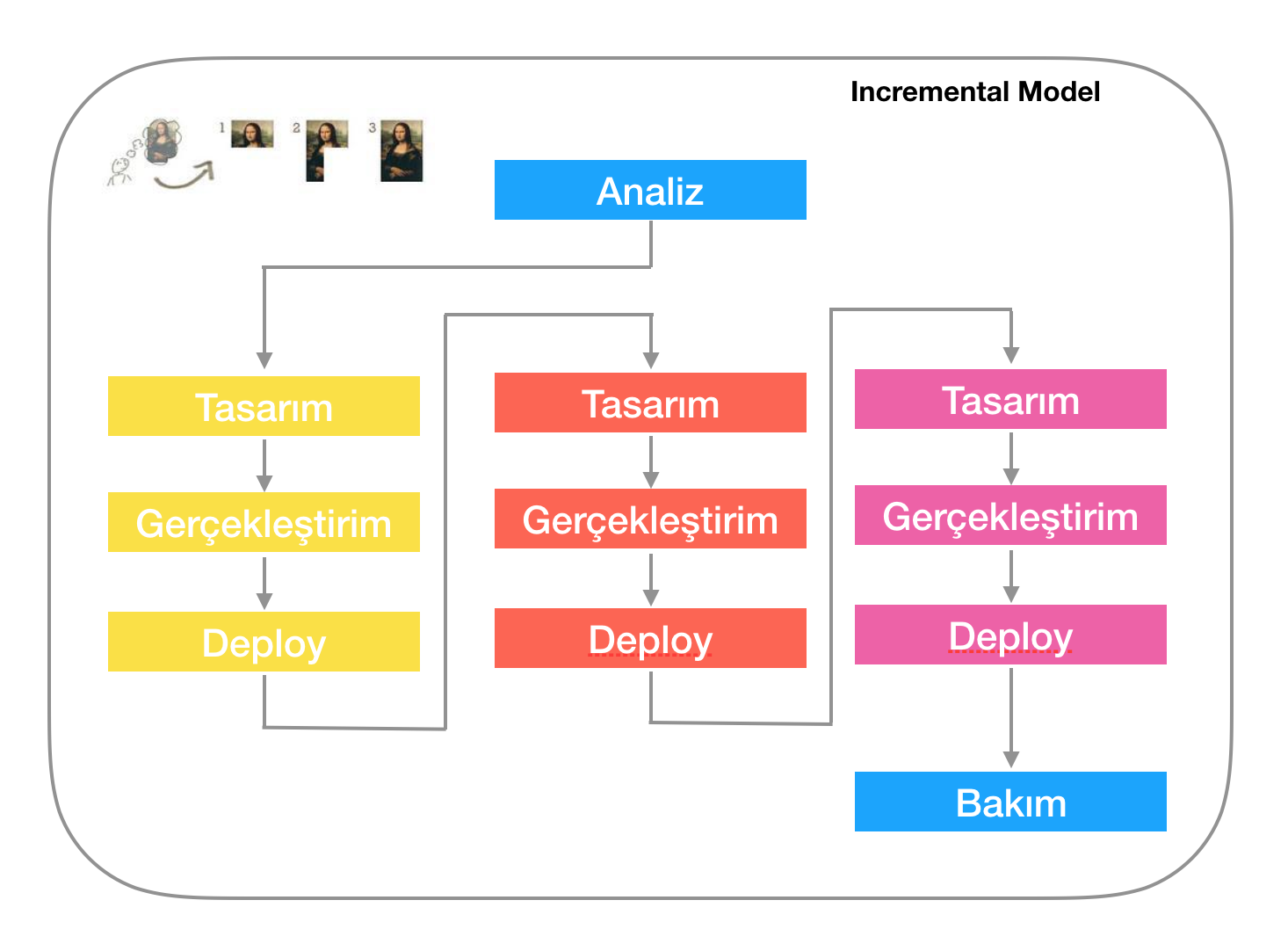
**Incremental (Arttırımlı) Model**

Proje Yöneticisi işi çalışabilen parçalara/modüllere ayırabilme zorunluluğu getirir. Bu gereksinimleri karşılayan projelerde Incremental Model kullanılarak geliştirme yapılabilir.

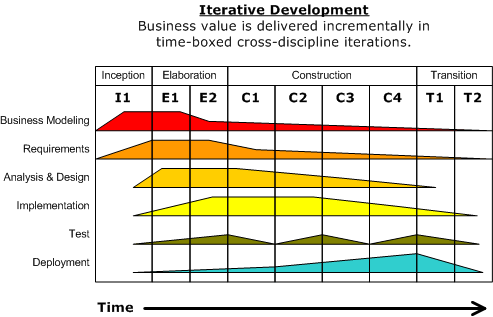
Sistem parçalara/modüllere ayrıldığında, sistemin tasarımını yapmaya gerek kalmadan kodlama, test ve deploy etme süreçleri tamamlanabilir ve müşteri ile daha kısa sürece iletişime geçilip görüşleri alınabilir. Bir bakıma küçük Waterfall modeli. Modelde döngü söz konusudur. Planlama aşaması ile başlar. Planlama tamamlandıktan sonra Analiz aşamasına geçilir ve döngüye girilir. Canlıya çıkılan programdan gelen bilgiler, gelen müşteri istekleri doğrultusunda planlamada değişiklikler yapılabilir.

Bu modelin başarısı bölünen parçaların birbirinden olabildiğince ayrı/soyut olması ve birbirleri ile entegrasyon ihtiyaçlarının az olması sayesinde olur. Aksi durumda bu model Waterfall Modelinden daha maliyetli bir hale gelir.

Bu modelin avantajları; başarısızlık riskinin az olması, sistemin fazla gözden geçirilmesi ve böl-yönet sistemlerde kullanılması olarak gösterilebilir. Dezavantajları ise; geliştirenlerin deneyimli olması gerekir, tanımlamaları iyi yapmak gerekir, tekrar etme yoktur.



**Rational Unified Process**



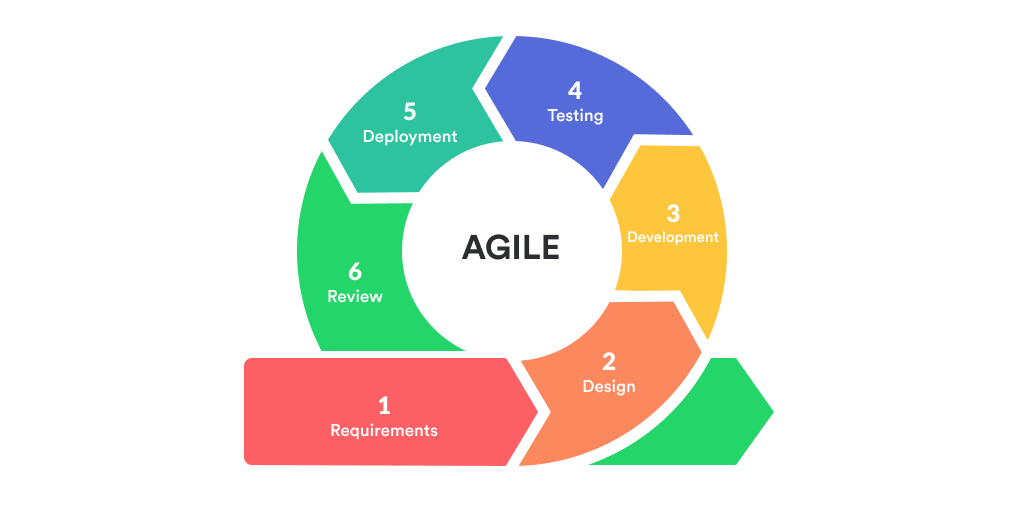
Rational Software firması tarafından geliştirilen bu süreç yapısı Iterative Geliştirme modeli üzerine kurulmuştur. Süreç genel olarak hangi aşamaya ne kadar efor, zaman harcanacağı üzerinde durulur. Bu modelde süreçler, temel süreçlerden farklıdır. İlerleyen zaman aralıklarında süreçler birleştirilmiştir.

* **Inception (Başlangıç) I1:** İş süreçlerinin belirlenip, hedeflerin neler olduğu ve gerçekten ihtiyaç var mı gibi sorulara cevap aranan aşamadır. Fizibilite çalışmaları bu aşamada yapılır. Bu aşamada Analiz & Tasarım ve diğer kodlama, Test ve Deploy işleriyle uğraşılmaz.
* **Eleboration (Detaylandırma) E1 – E2:** Eldeki iş modelini, analiz ve tasarımının detaylandırıldığı zaman aralığıdır. Gereksinimler belirlenir, Use Case’ler hazırlanır, Riskler belirlenir. Bu aşamada kodlama az da olsa başlar.
* **Construction (İnşa Etmek, Oluşturmak) C1 – C2 – C3 – C4:** Projenin en uzun aşamasıdır. Kodlama, Test ve Deploy işlemlerinin arttığı kısımdır. Birden fazla Iteration (yineleme) yapılır. Bazı durumlarda hata olması halinde Planlama ve Analiz kısmına geri dönülerek düzeltmeler yapılabilir.
* **Transiction(Geçiş) T1 – T2:** Ürünün canlıya alınıp müşteriye ulaştırılması aşamasıdır. Beta testleri ile sistemin sistem müşterinin isteklerine uyup uymadığı test edilir. Kullanıcı ve bakım ekiplerine eğitimler verilir.

Bu modelin ismi Unified Process olmasına rağmen aslında bir Framework. Bu Framework un bir parçası olarak diğer modeller kullanılabilir, her aşamada farklı yazılım süreçlerinden faydalanılabilir. Ayrıca bu süreç, mimariyi öne koyan bir yapıya sahip. Avantajları olarak; Uyarlanabilir, kaliteli ve tekrar kullanılabilir, risklere odaklanıp başarı şansının arttırması ve esnekliği diyebiliriz. Bu avantajlar sayesinde büyük projelerde kullanmak için uygundur. Dezavantajları ise; karmaşık bir yapısı, kaynak ihtiyacının fazla olması ve küçük projeler için çok fazla iş oluşturması diyebiliriz.

**Agile (Çevik) Model**

Agile (Çevik) Geliştirme Modeli her şeyden önce müşteri ile sürekli iletişim halinde olarak ve ayrıca ekip üyeleri arasındaki iletişime odaklanarak, katma değerli yazılımları sürekli teslimat yoluyla müşteri memnuniyeti aramamız gerektiğini savunur.



2001 yılında Kent Beck ve 16 arkadaşı tarafından **çevik manifesto** oluşturulmuştur. Üretilen bu modelleme biçimi, kapsadığı değerler, prensipler ve pratikler sayesinde *geleneksel modellere* göre yazılımlara daha esnek ve kullanışlı biçimde uygulanabilir. Bu manifestoda;

* Süreçler ve araçlar yerine **Bireyler ve Etkileşimler**
* Kapsamlı Belgeler yerine **Çalışan Yazılım**
* Sözleşme Görüşmeleri yerine **Müşteri İlişkileri**
* Plana Bağlı Kalma yerine **Değişime Karşılık Vermeye**

kanaat getirilmiştir. Özetle sol taraftaki maddelerin değerlerini kabul ederek, sağ taraftaki maddeleri daha değerli bulmaktadırlar.

Bu manifestonun altında yatan temel prensipler şunlardır;

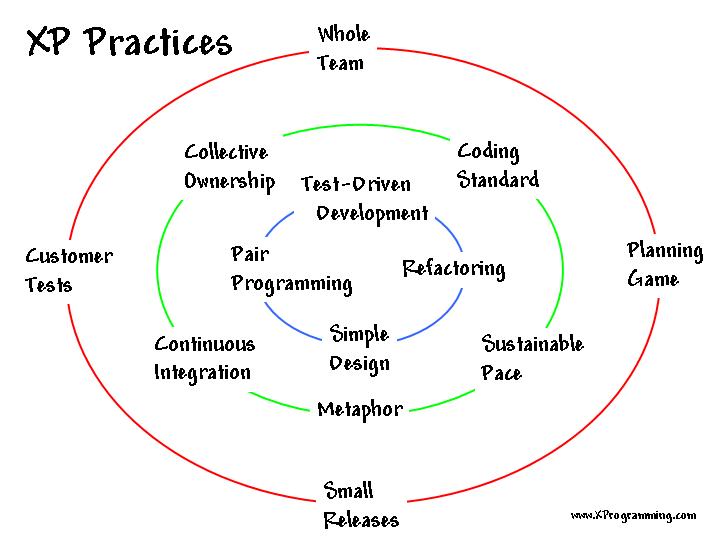
* Projelerin en kısa sürede ve devamlı şekilde teslim edilip müşteri memnuniyetinin sağlanması.
* Değişen süreçler projenin son aşamasında bile kabul edilmelidir. (Esneklik)
* Mümkün olan en kısa sürede, çalışan yazılım müşteriye teslim edilmelidir.
* Tüm ekip proje boyunca birlikte çalışmalıdır.
* Projede motive insanlar yer almalıdır. Onlara ihtiyaçları olan ortam ve destek sağlanmalı ve işi başaracakları konusunda motive edilmelidir.
* Ekipteki en etkili bilgi alışverişi yüz yüze olandır.
* Çalışan yazılım ilerlemenin birinci ölçüsüdür.
* Çevik süreçler sürdürülebilir geliştirmeyi teşvik etmektedir.
* Mükemmeliyet ve iyi tasarım konusundaki özen çevikliği arttırır.
* Sadelik ve basitlik önemlidir.
* En iyi mimariler ve tasarımlar kendini organize edebilen ekiplerden çıkar.
* Ekip düzenli aralıklarla nasıl daha etkili ve verimli olabileceğinin üzerinde düşünür ve davranışlarını buna göre ayarlar ve düzenler.

Çevik yazılım geliştirme modelinin avantajları; ekibin motivasyonunun yüksek olması, kısa sürede müşteri memnuniyetinin sağlanması, üretkenlik, yazılım kalitesinin artması ve maliyetlerin düşmesi diyebiliriz. Bu yöntem sonucunda projenin başarısı %55’e kadar artış gösterebilir.

**Çevik Yazılım Geliştirme Metotları**

* Extreme Programming (XP)
* Scrum
* Kanban
* Agile Unified Process,
* Feature-Driven Development (FDD)
* Test-Driven Development (TDD)
* LEAN Development
* Dynamic System Development Methodology (DSDM)
* Microsoft Solution Framework (MSF)

**Extreme Programming (XP)**

XP, Kent Beck tarafından 1999 yılında bir yazılım geliştirme disiplini olarak ortaya çıkarılmıştır.

XP, kolay, grup içi iletişime önem veren, geri dönüşlerin daha fazla olmasına imkân sağlayan yazılım geliştirme yöntemidir. 4 değer üzerine kuruludur. Bunlar; basitlik, iletişim, geri dönüşüm ve cesarettir.

**Basitlik:** Karmaşık çözümler XP’nin mantığına aykırıdır. Bu yüzden olabildiğince basit bir sistem geliştirilmelidir. XP basitliği sağlamak için günün ihtiyaçlarını hedef alarak, esnek ve basit sistem geliştirilmeye çalışılır.

**İletişim:** Genel olarak baktığımızda projelerde çıkan önemli problemler iletişim eksikliğinden kaynaklanır. Bu sebeple projenin başarıya ulaşabilmesi için iletişim çok önemlidir. XP bu iletişim eksikliğini ortadan kaldırmayı amaçlar. XP’de iletişim yüz yüze olmalıdır.

**Geri Bildirim:** Geri bildirimler çok önemlidir. Bunlar sayesinde sistemde ortaya çıkabilecek hatalar ortadan kaldırılır. (Yazılımcılar, sistemin Unit testlerini oluştururlar ve sistem hakkında somut bilgiler elde ederler.) Müşteri ekibin bir parçasıdır.

**Cesaret:** En zor kısımdır. Projedeki başarısızlıklardan korkmak yerine, başarısızlığı oluşturan nedenlerin üstüne gidilmelidir. Başarısızlıktan korkan bir ekipte proje geliştirmesi süresi yavaşlar. XP başarısızlıklardan korkmayı değil, en kısa sürede telafi etmeyi önerir.

**XP Pratikleri**

Yazılım geliştirmede kolaylığı ve esnekliği sağlamak için XP 12 farklı pratiği ön görür.

* Planlama Oyunu
* Ekipte Müşteri
* Önce Test
* Basit Tasarım
* Çiftli Programlama
* Sürekli Entegrasyon
* Yeniden Yapılandırma
* Kısa Aralıklı Sürümler
* Ortak Kod Sahiplenme
* Metafor
* Kodlama Standardı
* Haftada 40 saat

**Scrum**

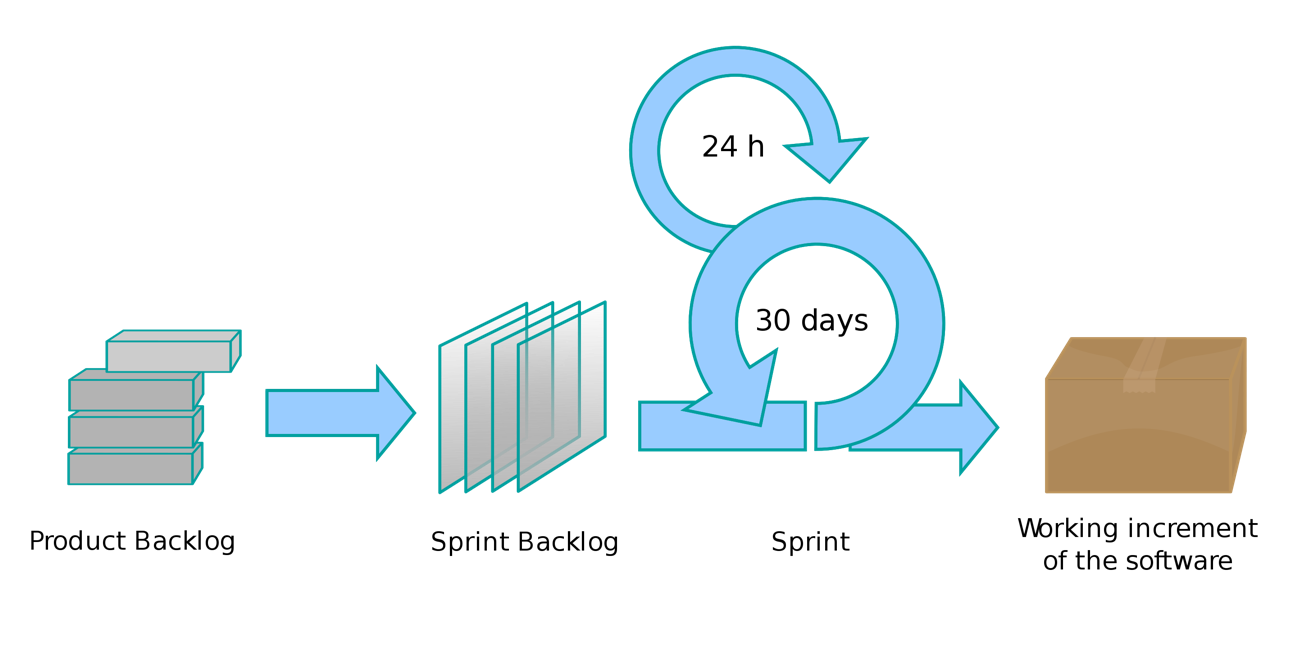
En çok kullanılan Agile geliştirme metotlarından biridir. Scrum, **sprint(**işlerin küçük birimlere bölünmesi**)** olarak bilinen geliştirme döngüleri veya aşamaları ve bir yazılım ürünü için geliştirme süresini maksimize edilmesiyle karakterize edilir. Genellikle yazılım ürünleri için proje yönetiminde kullanılır, ancak her alanda kullanılabilir.

Scrum yaklaşımında ekip çalışması çok önemlidir. Bu yüzden her gün “**Scrum Daily Meeting**” adı verilen 15 dakikalık toplantılar gerçekleştirilir. Bu toplantılar ile sürekli iş takibi sağlanır.

Product Backlog aşaması, müşteriler ile görüşülüp önceliklerin listelendiği süreçtir.

Sprint Backlog aşaması, 15 – 30 gün arasında sürmesi beklenen proje zaman dilimidir.

Sprint’in uygulama zamanı her zaman aynı olmalı ve uzatılmamalıdır. Bir Sprint en az bir hafta en fazla 4 hafta sürmelidir.



Scrum Süreci

Scrum avantajları olarak; takım günlük olarak toplantı yaptığından iş geliştirme motivasyonu fazla olur, şeffaflığı sayesinde ekibin tüm üyeleri veya bir kurum tarafından takip edilmesi kolaydır, kaliteye odaklanılır ve daha az hata olur, sprintler halinde işler bölündüğü için ekip üyeleri tamamlanmayan sprintleri takip etmesi kolaylaşır. Dezavantajları olarak; ekip üyelerinin rolleri kesin olarak tanımlanamayabilir, projenin bölümlere ayrılması bazen projeden sapmalara neden olabilir.

**Kanban**

Kanban, yazılım geliştirmede ne, ne zaman ve ne kadar üretileceği konusunda karar vermeye yardımcı olan görsel bir süreç yönetim sistemidir. Sistemin amacı; çok görevliliği azaltma, boşa geçen zamanı en aza indirme, müşterinin ihtiyaçlarını en öncellikli değerlendirmedir. Kanban sisteminde rollerden ziyade ihtiyaçların karşılanması, müşteri ve projeye odaklanılmaktadır.

Kanban’ın altı genel uygulaması vardır: görselleştirme, devam eden çalışmaları sınırlama, akış yönetimi, politikaları açıkça göstermek, geri bildirim döngüleri kullanmak ve işbirlikçi olmaktır.

Kanban avantajları olarak; projedeki tüm görevleri görme (tamamlandı, yapılıyor, yapılacak gibi), görevlerin sayısını sınırlandırabilme kolaylığı (yani işin miktarını, çözümünü veya teslim edilebilirliği göz önüne alınarak) ve kesintisiz teslimata izin vermesidir. Dezavantajları olarak; Ekip üyeleri Kanban panosunu yanlış anlayabilir (güncel olmaması gibi durumlar) ve panoda zaman dilimi olmadığı için aşamalarla ilişkili gecikmeler olabilir.



Basit Bir Kanban Panosu

## 200601073

Berkay Çoban

## KAYNAKÇA

<https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-fazlar%C4%B1-1f016ce1d412>

<https://medium.com/architectural-patterns/yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-modelleri-62915545c51e>

<https://furkanalniak.com/yazilim-muhendisligi-yazilim-yasam-donguleri/>

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

<http://ybsansiklopedi.com/wp-content/uploads/2015/08/Yaz%C4%B1l%C4%B1m-Geli%C5%9Ftirme-Modelleri-Yaz%C4%B1l%C4%B1m-Ya%C5%9Fam-D%C3%B6ng%C3%BCs%C3%BCSDLCYBS.pdf>

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

<https://resources.sei.cmu.edu/asset_files/SpecialReport/2000_003_001_13655.pdf>

<https://agilemanifesto.org/>

<https://medium.com/@gzbykyasin/agile-yaz%C4%B1l%C4%B1m-geli%C5%9Ftirme-2090d8baa25a>

<https://www.xpand-it.com/blog/top-5-agile-methodologies/>

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

<https://tr.wikipedia.org/wiki/Kanban_(geli%C5%9Ftirme)>

Doç. Dr. Deniz KILINÇ, Bakırçay Üniversitesi Yazılım Mühendisliği Temelleri Dersi 3. ve 4. Hafta Sunumları