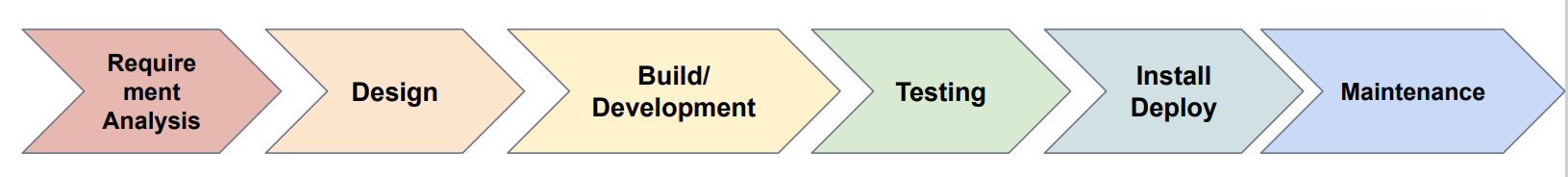
**Yazılım  Geliştirme Yaşam Döngüsü  (SDLC)**

Yazılım  **Geliştirme Yaşam Döngüsü** (SDLC), yazılım geliştirmek için kullanılan bir süreçtir. SDLC, yüksek kaliteli bir ürün tasarlamak ve geliştirmek için kuruluşun takip etmesi gereken adım adım bir prosedürdür. Yazılım geliştirme yaşam döngüsünün aşamaları, belirli bir yazılımın nasıl geliştirileceğini ve sürdürüleceğini açıklar. Yaşam döngüsü, kaliteli bir ürün/yazılım geliştirmeyi amaçlar. SDLC, müşteri gereksinimlerine göre çalışıp çalışmadıklarını kontrol etmek için gözden geçirilebilen ara ürünler üretir. Bir SDLC modeli, bir yazılım geliştirme projesinde her aşamada gerçekleştirilen faaliyet türlerini ve faaliyetlerin mantıksal ve kronolojik olarak birbirleriyle nasıl ilişkili olduğunu açıklar.

***“Özet olarak SDLC yazılım geliştirmek için kullanılan bir süreçtir.”***

SDLC’ nin tam net olarak kabul edilen safha sayısı yoktur kaynaklara göre 5 ile 10 arasında farklılık göstermektedir ancak kabul edilmesi gereken safha sayısı 6 olarak belirtilebilinir.



**Requirement Phase:** Gereksinimlerin analiz edildiği ve toplanıldığı aşamadır. En önemli aşamadır eğer gereksinimler doğru bir şekilde analiz edilmez ise doğru bir sonuç elde edemeyiz. Bu safhada SRS diye bir doküman hazırlanır. Doküman çok önemlidir müşteri ve yazılım geliştiricisi arasında sözleşme gibi düşünülebilinir. Dökümanı hazırlayan meslek grubu ise iş analisti olarak adlandırılır.

**Design Phase:** Hangi yazılım dili kullanılacak, database ne olacak, domain ne olacak, hangi tier da olacak bu tarz olayların belirlendiği aşamadır tüm bunlar SRS dökümanındaki istemler göz önüne alarak mimari yapı orataya çıkarılır. Bu mimari proje hazırlanmasını sağlayan meslek grubu ise solution architecture veya software architect

**Build / Development Phase:** Bu aşamada developer ekibi projenin sourch kodlarını yazmaya başladıkları aşamadır. Tüm kodlar developer ekibindeki kişilerin sorumlu olduğu kısımları yazmasıyla devam eder.

**Testing Phase:** Kod doğru çalışıyormu, Requirement kısmındaki istekleri karşılıyormu, bug varmı gibi testleri bulunduğu aşamadır. Testing ekibi tarafınca tüm bu işlemler yürütülür.

**Deploy / Deliver Phase:** Kodun büyük bir kısmı test edilmiş olsa bile test edilmemiş kısımda kalan hatalar kullanım esnasında ortaya çıkabilir. Bu hataların giderildiği, çözüldüğü aşama ise deploy aşamasıdır.

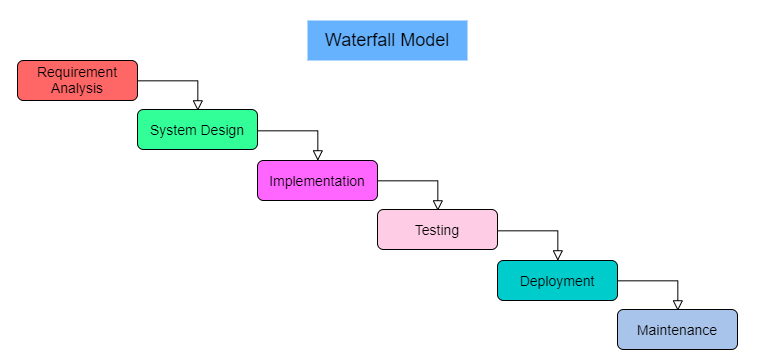
**Maintenance Phase:** SRS dökümanında belirtilen süre boyunca uygulamanın kullanırken ortaya çıkabilecek düzeltmeler ve destek gibi hizmetlerin sağlandığı süreçtir.

Yukarda belirttiğimiz üzere SDLC’ nin 6 aşaması bulunmaktadır. Birden fazla modellerin ortaya çıkış sebebi ise zamanla yazılım gelişterme alanındaki ihtiyaçlara daha fazla yanıt veremez olmalarıdır. Aşağıda Çeşitli Yazılım geliştirme modelleri veya metodolojileri bulunmaktadır.

* V-Model
* Agile Model
* Spiral Model
* RAD Modeel
* Iterative Model
* DevOps Model
* Waterfall Model

Bu metodolojilerden en önemli olanı bizim için Agile modeldir. Agile modeline ayrı bir ders olarak değineceğiz. DevOps ve Waterfall modellerini ise bilgi sahibi olmak ve Agile model arasındaki farkları daha net benimsemek için değineceğiz. Günümüzde yazılım alanındaki en çok kullanılan SDLC modeli % 90 oranıyla Agile’ dir.

**Waterfall Model**

**Şelale Modeli**, bazı tür projelerin başarısını sağlamak için yazılım geliştirmede kullanılan ilk SDLC modelidir. Bu modeli anlamak ve kullanmak çok basittir. Model, yazılım geliştirme sürecinin aşamaları arasındaki geçişi bir şelale gibi şekillendirmiştir. **Doğrusal sıralı** bir akışta yazılım geliştirme sürecini gösterir . Bu, her aşamanın bir sonraki aşamanın başlangıcından önce tamamlanması gerektiği anlamına gelir.  **Şelale** yaklaşımında, tüm yazılım geliştirme süreci ayrı aşamalara bölünmüştür. Genellikle bu modelde, bir katmanın sonucu sıralı olarak bir sonraki katmanın girdisi olarak işlev görür.

**Requirement Analysis:** Bu süreç sırasında, oluşturulacak tüm olası sistem gereksinimleri toplanır ve bir gereksinim belirtim belgesine kaydedilir.

**System Design:**  Bu aşamada, ilk aşamadaki gereksinimler gözden geçirilecek ve cihaz tasarımı hazırlanacaktır. Bu sistem tasarımı, donanım ve sistemin özelliklerini belirlemeye ve genel sistem mimarisini tanımlamaya yardımcı olur.

**Implementation:** Sistem ilk olarak, sistem tasarımından gelen girdilerle, bir sonraki aşamaya entegre edilen birimler adı verilen küçük programlarda geliştirilir. Her birim, Birim Testi olarak bilinen işlevleri için geliştirilir ve test edilir.

**Integration and Testing:** Her bir ünite test edildikten sonra, uygulama sürecinde oluşturulan tüm üniteler bir sisteme dahil edilir. Tüm sistemin entegrasyonu sonrası eksiklik ve eksiklikler kontrol edilir.

**Deployment of system:** Test yapıldıktan sonra, ürün müşteri ortamında dağıtılır veya piyasaya sürülür.

**Maintenance:** İstemci ortamında bazı sorunlar ortaya çıkar. Bu sorunları gidermek için yamalar yayınlandı. Ürünü geliştirmek için bazı geliştirilmiş sürümler yayınlanmıştır. Bu tür iyileştirmeleri yapmak için tüketici ortamında bakım yapılır.

***Şelale Modelinin başlıca avantajlarından bazıları şunlardır;***

* Her geliştirme aşaması katı bir sırayla ilerler.
* Kullanımı ve anlaşılması kolay ve basit.
* Test aşamasındaki hata sayısı çok azdır.
* Görevlerin düzenlenmesi kolaydır.
* Sürecin ve sonuçların iyi bir dokümantasyonu var.

***Şelale Modelinin önemli dezavantajlarından bazıları şunlardır;***

* Çok fazla düşünmeye veya gözden geçirmeye izin vermez.
* Yüksek miktarda risk ve belirsizlik vardır.
* Aşamalardaki ilerlemeyi ölçmek zor olabilir.
* Ürün tüm aşamaların sonuna kadar beklemek zorundadır.
* Değişen gereksinimlere uyum sağlayamaz.