YAZILIM MÜHENDİSLİĞİ TEMELLERİ

HAZIRLAYAN: MİRAÇ ZORLU

ÖĞRENCİ NUMARASI: 220601121

ÖĞRETİM GÖREVLİSİ: ZEKERİYA ANIL GÜVEN

Yazılım Yaşam döngüsü

Yazılım Geliştirme Yaşam Döngüsü bir yazılım yöntemler bütünü yaklaşımıdır.

Bir sarmal hareket halinde olan bir yaklaşımdır. Aşamaları aşağıdaki şekilde sıralanabilir.

1. Planlama

2. Tanımlama

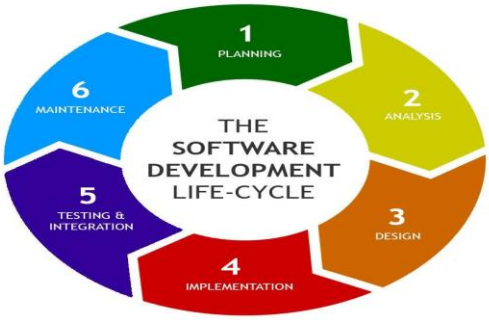
3. Tasarım

4. Geliştirme

5. Entegrasyon ve testler

6. Uygulama

7. Bakım



Hayatın bir döngü içinde döndüğü gibi, yazılımın da kendi doğal yaşam döngüsü vardır ve doğru analiz edilip kontrolleri yapıldığında başarıya götürebilir. Bu yaşam döngüsüne "sistem geliştirme yaşam döngüsü" denir ve herhangi bir sistem geliştirirken dikkate alınması gereklidir. Bu yazımızda SDLC adımları ayrıntılı olarak açıklayacak, avantaj ve dezavantajlarını anlatacağız ve literatürdeki diğer yazılım geliştirme yöntemleri ile karşılaştırılacağım.Proje yapımının önemli bir aşaması olan tanımlama, planlama sürecinden sonra gerçekleştirilir. Bu aşamada, projenin temel amaçları ve kavramları üzerinde konuşulur ve belirlenir. Bu aşama, projenin başarılı bir şekilde tamamlanabilmesi için oldukça önemlidir çünkü farklı insanlar arasında aynı fikirden bahsedilmiyorsa, farklı sonuçlara ulaşılabilir.

TANIMLAMA AŞAMASI

Tanımlama aşaması, bir analiz aşaması olarak da düşünülebilir. Bu aşamada, projenin problemleri ve ihtiyaçları tespit edilir ve bunlar üzerinde çalışılır. Böylece, projenin yaşam döngüsü ve/veya yazılımı tanımlanır. Ayrıca, projede nelerin istendiği konusunda analiz çalışmaları da yapılır.Bu aşamada, projenin tasarımı üzerine çalışılır. Tanımlanan amaçlar ve kavramlar doğrultusunda, projenin nasıl yapılacağına dair bir plan hazırlanır. Bu plan, projenin başarılı bir şekilde tamamlanabilmesi için oldukça önemlidir.Sonuç olarak, tanımlama aşaması, projenin temel amaçlarının ve kavramlarının belirlendiği ve analiz çalışmalarının yapıldığı bir süreçtir. Bu aşama, projenin tasarımı için temel oluşturur ve başarılı bir şekilde tamamlanabilmesi için gereklidir.

TASARIM AŞAMASI

Tasarım aşaması, yazılım veya sistem için bir planın çizildiği ve kararların alındığı bir süreçtir. Bu aşama, planlama ve tanımlama aşamalarına dayanır ve yazılımın ekranları, bileşenleri ve modülleri gibi detaylı özellikleri tasarlanır.Bu aşamada, yazılımın kullanıcı arayüzü ve fonksiyonelliği hakkında kararlar verilir. Hangi ekranların olacağı, ekranlarda hangi özelliklerin yer alacağı, nasıl geçişler olacağı gibi detaylı konular tasarlanır.Tasarım aşamasında, tüm kararlar verilir ve geliştirme sürecine geçilmeden önce herhangi bir soru veya karar bırakılmaz. Bu sayede, geliştirme aşamasında herhangi bir kararsızlık veya belirsizlik olmaz.Sonuç olarak, tasarım aşaması, yazılım veya sistem için detaylı bir planın çizildiği ve tüm kararların alındığı kritik bir süreçtir.

GERÇEKLEŞTİRME AŞAMASI

Projelerin başarılı bir şekilde tamamlanabilmesi için tanımlama ve planlama süreçleri oldukça önemlidir. Tanımlama aşaması, projenin temel amaçlarının ve kavramlarının belirlendiği, analiz çalışmalarının yapıldığı ve projenin tasarımının planlandığı bir süreçtir. Bu aşama, farklı insanlar arasında aynı fikirden bahsedilmesini sağlar ve projenin yaşam döngüsü veya yazılımı tanımlanır. Tanımlama aşaması, projenin başarılı bir şekilde tamamlanabilmesi için temel oluşturur.Geliştirme aşaması, yazılım projelerinde kodlama sürecinin başladığı veya sistem örneklerinin ortaya çıkmaya başladığı aşamadır. Bu aşamada, tasarım aşamasında alınan kararlar doğrultusunda projenin gerçekleştirilmesine başlanır.Geliştirme aşaması, inşaat projesinin mimari planının tamamlandığı ve gerçek inşaatın başladığı aşamaya benzetilebilir. Yazılım projelerinde, kodlama işlemi bu aşamada gerçekleştirilir.Özetle, geliştirme aşaması, yazılım projelerinde tasarım aşamasında alınan kararlar doğrultusunda projenin kodlandığı ve gerçekleştirildiği kritik bir süreçtir.

UYGULAMA AŞAMASI

Uygulama Proje artık yaşayan bir yazılım haline gelir. Yazılım kullanılır ve problemler ortaya çıkar. Problemlere karşılık yeni fikirler ortaya çıkar, yeni ihtiyaçlar oraya çıkar ve bunlar değerlendirilmeye alınır. Bu değerlendirmeler sonrasında bakım aşamasına geçilir.

BAKIM AŞAMASI

Bakım, yazılım projesinin ürün olarak sunulduktan sonra sürdürülmesi ve güncellenmesi anlamına gelir. Yazılımın yayınlanmasından sonra, zaman içinde hatalar ve sorunlar ortaya çıkabilir. Bunlar düzeltilmeli ve yazılım güncellemeleri yayınlanmalıdır.Bakım ayrıca yazılımın işlevselliğinin sürekli olarak korunmasını da içerir. Kullanıcıların geri bildirimlerine göre, yazılımın performansı ve kullanıcı deneyimi iyileştirilmelidir. Ayrıca, yazılımın çalıştığı donanım ve işletim sistemleri gibi diğer faktörlere uyumlu olması da önemlidir.Bakım aşaması, yazılım projesinin ömrünün sonuna kadar sürebilir. Bazı durumlarda, yazılım projesinin devam etmesi için daha büyük bir revizyon veya güncelleme gerekebilir. Bu nedenle, bakım, yazılım projesinin ömrü boyunca sürekli bir süreçtir.

SDLC'nin yazılım geliştirme aşamasında bize sunduğu avantajları şu şekilde sıralayabiliriz:

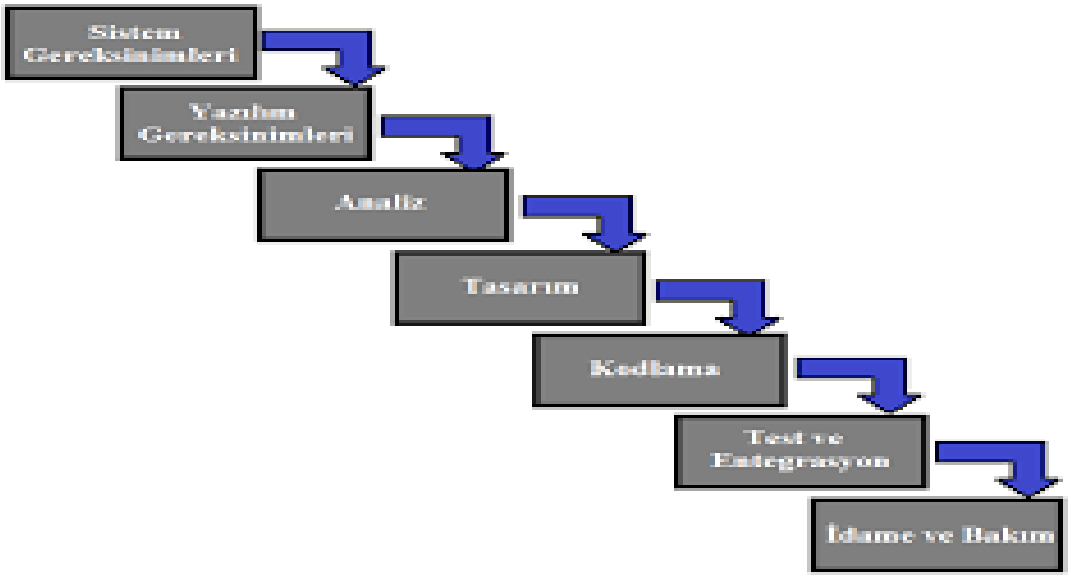
- Yüksek düzeyde yönetim kontrolü ve dokümantasyon sağlar.
- Yazılım geliştiricisinin süreç boyunca planlı bir şekilde ilerlemesini sağlar.
- Projede çalışan herkesin gerekli maliyet ve kaynakları anlamasını sağlar.

Yazılımın geliştirme sürecinde SDLC'nin yol açabileceği dezavantajları şu şekilde sıralayabiliriz:

- Proje geliştirme sürecini yavaşlatabilir.
- Projenin son aşamalarında düşük esneklik sağlar.
- Test aşaması geliştiriciler için çok karmaşık olabilir.

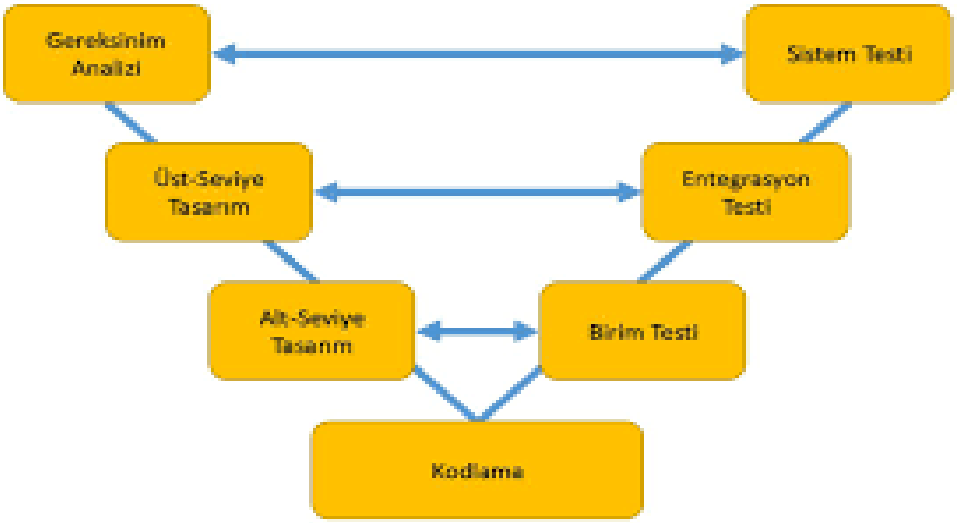
ŞELALE MODELİ

"Şelale Modeli" yazılım geliştirme sürecinde kullanılan bir yöntemdir. Bu model, geliştirme sürecinin adımlarının tek bir yönlü bir akışta sıralandığı ve her adımın tamamlanmadan bir sonraki adıma geçilemeyeceği bir modeldir. Bu modelin adı, su şelalelerinin akışı ile benzerlik gösterir. İlk adım, yazılımın tasarımıdır ve bu adımdan sonra kodlama, test etme, entegrasyon ve bakım gibi adımlar sırasıyla gerçekleştirilir. Bu süreç boyunca, her adım tamamlanmadan bir sonraki adıma geçmek mümkün değildir.



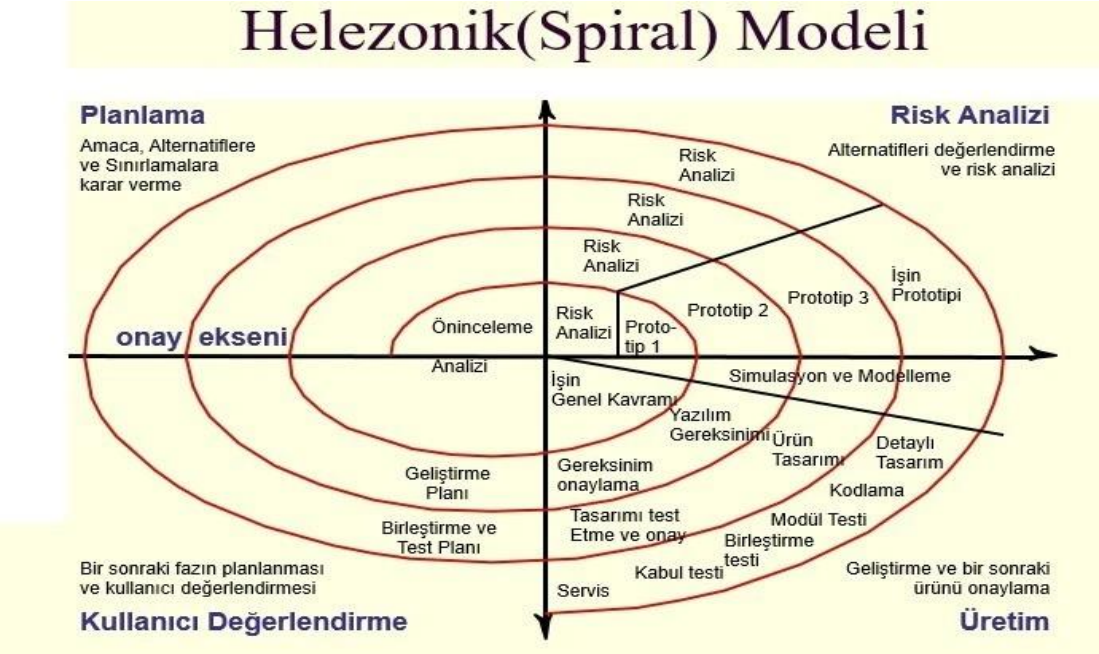
V MODEL

V-model şelale modelinin gelişmiş hali olarak düşünülebilecek bir yazılım geliştirme süreci sunar. Doğrusal bir yönde ilerlemek yerine, süreç adımları kodlama evresinden sonra yukarıya doğru eğim alır ve tipik V şeklini oluşturur. V-Model geliştirme yaşam çevriminin her bir evresi arasındaki ilişkileri gösterir. Yatay ve dikey açılar zaman veya projenin tamamlanabilirliğini ve soyut seviyeyi gösterir.



HELEZONİK MODEL

Yoğun müşteri katılımının görüldüğü bir modeldir ve çok kapsamlı risk değerlendirmesine odaklanır.Yineleme ve şelale modelinin bir kombinasyonudur**.** Bu gelişim sürecinde sürekli işlevsellik eklendiği için spiraldir.Her aşama bir tasarım hedefi ile başlayıp müşteri incelemesi ile biter. Yani her aşama küçük gereksinimler setidir. Bu müşteri incelemeleri güçlü bir onay modeli ve dokümantasyon kontrolü sağlar. Geliştiricilerin herhangi bir aşamada kodda değişiklik yapmasına olanak tanıdığı için belirsiz veya karmaşık özel, büyük ve kritik özellikteki projelerde tercih edilir. Spiral model 4 adımdan oluşur; Maliyet ve kaynak tahminleri kapsamlı bir *planlama,* stratejik değerlendirme ve risk analizi**,** yazılım ve test aşamasında prototip oluşturma, daha önce teslim edilen parçaların müşteri değerlendirmesi ve bir sonraki spirale geçmeden önce geri bildirim. Bu tekrarlanan spiral döngü, proje zaman çizelgesini uzatır. Maliyet bu modelin dezavantajıdır. Küçük projeler için kullanılmaz. Spiral Model kullanım alanları; belirsiz iş ihtiyaçları olan projeler, inovatif ihtiyaçları olan projeler, büyük ve karmaşık projeler, Ar-ge faaliyeti vb.



SCRUM MODELİ VE TERCİH NEDENLERİ

SCRUM, 1990'ların ortalarında Jeff Sutjerland ve Ken Schawaber tarafından geliştirilen, çevik yazılım geliştirme metodolojileri ile uygulanabilen bir proje yönetimi yaklaşımıdır. Yani SCRUM sadece yazılım geliştirmede değil her alanda uygulanabilir. SCRUM'da büyük projelerin parçalara ayrıldığı ve her parçaya "sprint" adı verilen ve ardından her sprint'in kademeli olarak geliştirildiği bir proje yönetimidir.SCRUM, gereksinimlerin kolayca tanımlanamadığı ve kaotik durumların beklendiği projeler için en uygun metodolojidir.Bu metodolojide bir iterasyonun tamamlanması 30 günü geçmiyor ve sürekli çalışma günlük 15 dakikalık toplantılarla takip ediliyor. Bu, geliştiriciler arasındaki iletişimi her zaman

güçlendirip sıkılaştırır ve her geliştiricinin projedeki her bir kişi hakkında bilgi sahibi olmasını ve ayrıca işlerin takip edilmesini sağlar.

SCRUM un günümüzde oldukça popüler ve en çok kullanılan yazılım geliştirme yöntemidir. Nedenleri ise şöyle sıralanabilir:

- Ekip içerisindeki iletişim düzeyinin yüksek olmasını sağlaması ve bu sayede hataların erken fark edilip düzeltilmesi,
- Kullanıcıdan sürekli olarak geri bildirim gerektirmesi ve bu geri bildirim ile beraber sorunların çözülüp azaltılabilbesini sağlaması,
- Diğer yazılım geliştirme metodolojileri gibi yinelemeli ve sürekli aktif olması,
- Değişen gereksinimlere hızlı bir şekilde tepki verilmesine ortam hazırlamış olması,
- Büyük parçaları küçülterek hızlıca ilerlemeyi sağlayan bir yapıya sahip olması.
- Proje süresince geçen zamandan ve paradan büyük ölçekte tasarruf edilmesi,
- Her teknolojiye uyum sağlayabilip. Teknolojideki son gelişmelere uyum sağlayabilmesi,
- Karmaşık olan ve gereksinimleri tam belirlenememiş projeler için ideal bir yapı olması

BAZI DÖNGÜ MODELLERİNİN KARŞILAŞTIRMASI

No	**Model/Özellik**	**Şelale modeli**	**V modeli**	**Helozonik**	**scrum**
1	**Gereksinin Belirleme**	Başlangıç	Başlangıç	orta	ileri
2	**Maliyet**	Düşük	Maliyetli	Maliyetli	Düşük
3	**Başarı Garantisi**	Düşük	Düşük	Orta	-
4	**Uzmanlık Gerekliliği**	Düşük	Orta	Orta	-
5	**Fazların Örtüşmesi**	Hayır	Hayır	Hayır	Hayır
6	**Maliyet Kontrolü**	Hayır	Evet	Evet	Evet
7	**Basitlik**	Basit	Basit	Orta	Karmaşık
8	**Risk Duyarlılığı**	Yüksek	Yüksek	Yüksek Değil	Yüksek

9	**Esneklik**	Katı	Katı	Düşük	Düşük