**Yazılım Yaşam Döngüsü ve Modelleri**

**Çevik Yazılım Geliştirme ve Scrum**

**Yazılım Yaşam Döngüsü Nedir?**

Her varlığın sahip olduğu bir yaşam döngüsü varken elbette yazılımlar da bir yaşam döngüsüne sahiptir. Bir yazılım elbette kodlanarak meydana getirilir ancak sadece kodlamak yetmez. İnsanoğlunun hayatını kolaylaştırmak için meydana getirilmekte olan yazılımlar, yine insanoğlunun değişen ihtiyaç/isteklerine göre birtakım değişimler geçirmek durumunda kalır. Bu nedenle kodlandıktan sonra bile geliştirilmeye devam eder. Yazılımlar oluşturulurken her biri birbirinden farklı belirli aşamalardan geçer, oluşturulduktan sonra da birtakım işlemler görür ta ki yazılımın kullanımdan kaldırılıp (güncel ihtiyaçları karşılayamama, talebin azalması vb. durumlardan dolayı) yerine başka bir yazılım getirilene dek. Yazılım Yaşam döngüsü, bu işlemlerin hepsini ifade eden modellemeleri içerir. Yazılım Yaşam Döngüsü, uygulanan teknikler, aşama sayısı ve yazılımın yerine getireceği görev gibi durumlara göre farklı biçimde modellere sahip olur.

**Yazılım Yaşam Döngüsü Adımları Nelerdir?**

**Gereksinim Analizi (Requirements Analysis):**

Yazılım

Yaşam

Döngüsü

Yazılım geliştirilmesi için gerekli olan hem zaman hem de parasal yönden maliyetin, müşterinin ihtiyaçlarının ve kullanılacağı alana göre hangi yazılım yaşam döngüsü modelinin kullanılacağı belirlenir.

**Tasarım (Design):**

Yazılımı kodlamak için daha önceki adımlarda belirlenmiş tasarım kalıplarıyla yazılım mimarisi oluşturulur.

**Uygulama/Kodlama (Implementation/Coding):**

Oluşturulmuş yazılım mimarisine göre yazılım kodlanır.

**Test Etme (Testing):**

Tamamlanmış yazılım görevini eksiksiz ve sorunsuz yerine getirme bakımından test edilir ve bunun sonucunda karşılaşılan eksikler/sorunlar için kodlama ekibine geri bildirimler sağlanır.

**Bakım (Maintenance):**

Kullanıma çıkarılmış yazılımın bakımı yapılır. Zamanla kullanıcıların karşılaştığı sorunlar ve gereksinimler doğrultusunda ihtiyaç duyulan güncellemelerin gerçekleştirildiği kısımdır.

**Yazılım Yaşam Döngüsü Modelleri Nelerdir?**

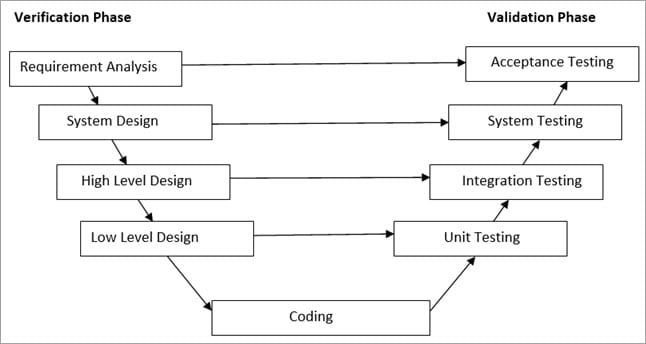
**Çağlayan Döngü:**

En ilkel modellerden birisi olarak kabul edilir. Bu modelde bir adım bitirilmeden diğer adıma geçilmez, sıra sıra her adım tamamlanır. Bu nedenler modelin işleyiş halindeyken yönetimi kolaydır. Aşamaların sırasıyla tamamlanmasından dolayı gereksinimlerin analizi ve planlama aşaması büyük önem arz etmektedir. Gereksinimlerin sonradan değişmeyeceği projeler için ideal model olduğu gibi aksi projeler için bu model gerek zaman bakımından gerekse parasal bakımdan büyük zararlara neden olabilir.

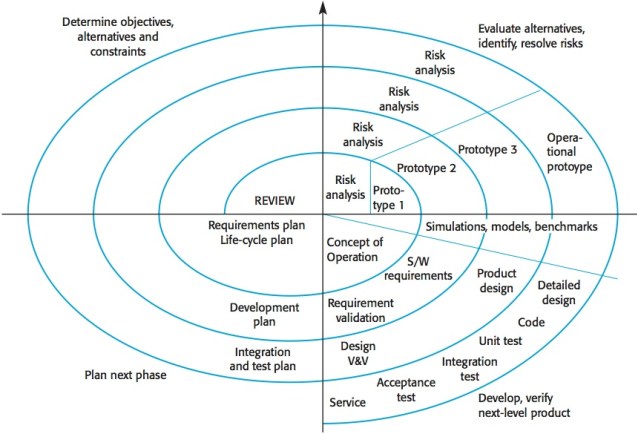
**Barok Modeli:**

Bu modelde aşamalar arasında nasıl geri dönüş yapılacağı belirli olmamakla birlikte aşamalar doğrusal bir şekilde bulunur. Dokümantasyon işlemleri yazılım geliştirildikten sonra tamamlanır ve diğer aşamalardan bağımsız olarak ele alınır. Eskiden kullanımı sık olsa da günümüzde o kadar revaçta olmayan bir modeldir.

**Big Bang Modeli:** Bu model, her ne kadar kısa ve basit projeler için kullanışlı olsa da diğer projeler için başarı olasılığı düşük bir modeldir. Genellikle küçük bir geliştirici grubunun veya bir geliştiricinin kullandığı yöntemdir. Geliştirme ve test kısımlarından oluşur ve dokümantasyon aşaması bu modelde yer almaz. Planlama ve analiz aşamaları geliştirici/geliştiriciler ‘in kendi anlayışına göre gerçekleştirilir. Projede görevli fazla kişi olmaması ve modelin işleyişinin basit olması sebebiyle yönetimi basit olan bir modeldir.

**V modeli:**

Tıpkı waterfall modelinde olduğu gibi bir adım tamamlanmadan diğer adıma geçilmez. Bunun yanında her aşama tamamlandıktan sonra o aşama teste tabi tutulur. Bundan dolayı bu modelde kullanıcı aktif rol oynar. Bu yönüyle başarı oranı waterfall modeline göre oldukça yüksektir. Dezavantajlarından bazıları ise zaman ve maddi bakımdan maliyetlidir ve sonradan değişen gereksinimlere sahip projelerde tamamlanmış aşamalarda değişiklik yapılması gerekeceğinden maliyet artışına sebep olur.

**Spiral Model :** Yazılım yaşam döngüsü adımlarını belirli fazlar halinde tekrar tekrar tamamlanır. Dört kısma sahiptir: gereksinim analizi, risklerin tanımlanması ve çözülmesi, geliştirme aşaması ve bir sonraki aşamanın planlaması. Her faz tamamlandıktan sonra o faza ait prototip üretilir. Bu nedenle yazılıma getirilecek güncellemeler için yapılan değişiklikler kolay olur. Adından anlaşılacağı gibi Spiral şeklinde bu işlemleri tekrar eder. Böylelikle müşteri ihtiyaçlarını karşılama açısından yüksek oranda başarı gösterir. Uzun zaman sarf edilmesi ve birçok aşamaya sahip olduğundan dolayı proje yönetiminin zor olması dezavantajları arasındadır.

**Kodla Düzelt Modeli:** Uygulanması basit ama bir o kadar da maliyetli olabilecek bir modeldir. Nedeni ise dokümantasyon olayı bu modelde yer almaz ve direkt kodlama kısmına geçilir, dokümantasyon olmadığı için yazılıma daha sonradan getirilecek güncelleştirmeler geliştirici ekip için pahalıya mal olur. Kısa projeler için uygulanabilecek bir modeldir fakat büyük projeler için elverişsizdir.

**Artımsal Geliştirme Modeli :** Gereksinimler bu modelde odak noktasıdır. Bu nedenle gereksinimler birbiriyle olan ilişkileri ve önem sırasına göre parçalara ayrılıp sırasıyla tamamlanır. Bir gereksinim için çözüm işleme koyulduğunda diğer gereksinimler arka planda kalır. Geliştirilen yazılım sürümü bir önceki sürümü kapsayacak biçimde ve dolayısıyla işlev sayısı artmış bir durumda geliştirilir. Kullanım ve geliştirim olayları paralel olarak devam eder.

**Hangi Model Hangi Projede Kullanılır?**

Her model her projeye uygulanabilir mi? Kesinlikle uygulanamaz değildir elbette. Ancak müşteri gereksinimlerini karşılayamayız ve/veya maliyetli olur. Çünkü modeller kullanıldığı projeye göre verimlilik sağlar. En çetrefilli proje için en kolay modeli uygularsak geliştirme sürecinde kısmen zaman ve para bakımından maliyet azalır fakat teslim sonrası süreçlerde büyük sorunlarla karşılaşılması kaçınılmazdır. Aynı şekilde en basit proje için uygulanması en zor modeli seçebiliriz ve bu sefer kusursuz bir sonuç ortaya çıksa bile maliyet yüksek olur kısacası attığımız taş ürküttüğümüz kurbağaya değmez. Bu yüzden modelleri projelere göre kullanmak gerekir. Modelleri kullanıldığı proje türlerini belirtecek olursak. Kısa ve basit projeler için, Big Bang, Kodla Düzelt ve Barok modeli kullanılır. Uzun ve kompleks projeler için, Çağlayan Döngü ve Artımsal Geliştirme modeli kullanılır. Uzun, gereksinimlerin değişme durumu yüksek , teslim edildikten sonra sık sık güncellemeler alması gereken ve gereksinimlerin tam olarak belli olmadığı projeler için Spiral ve V modelleri kullanılır.

Projenin gereksinimlerine göre uygun Yazılım Yaşam Döngü modelinin tercih edilmesinin bu konudaki başarıyı arttırdığı bir gerçek. Bu başarı oranını daha da arttırmanın bir diğer yolu ise, projeye daha küçük projelere ayıran, geliştirme ekibinin sürekli iletişim halinde olmasını sağlayan ve müşteri ve kullanıcıyı geliştirme sürecine dahil eden Çevik Yazılım geliştirme metotlarına başvurmaktır. Öyleyse bir de Çevik Yazılım Geliştirmenin ne olduğuna göz atalım.

**Çevik Yazılım Geliştirme Nedir?**

Çevik yazılım geliştirme metotlarına göre bir projeyi daha küçük projelere ayırarak tamamlaması bu konuda büyük yarar sağlamakta. Her küçük proje tamamlanması sonucu oluşan ürün müşteriye sunulur ve böylece eksikler ve yeni gereksinimler saptanmış olur. Ayrıca, kapsamlı projeler için ekip çalışmasını kolaylaştıran, hem parasal bakımdan hem de zamansal bakımdan maliyeti oldukça azaltan, müşteri gereksinimlerini ve/veya isteklerini eksiksiz karşılayan ve projenin başarı durumunu arttıran metotlar bütününü barındırır. Süreçler ve araçlardan ziyade geliştirici ekipteki bireylerin iletişimine önem vererek ekibin motivasyonunu ve performansını olumlu yönde etkiler. Kapsamlı dokümantasyon yerine kademe sonunda meydana gelen çalışan yazılıma önem verilir. Böylelikle eksiklikler kolayca saptanır. Müşteri ile pazarlık yapmak yerine müşteri ile iş birliği içinde olmaya önem verilir. Son olarak sabit bir planı takip etmektense değişen ihtiyaçlara göre esnek bir çalışma stili benimsenir.

**Çevik Yazılım Geliştirmede Kullanılan Metodolojiler Nelerdir?**

Yazılım Yaşam Döngülerinde olduğu gibi Çevik Yazılım Geliştirme için de farklı yöntemler vardır. En yaygın yöntemler ise şunlardır:

* Extreme Programming (XP),
* SCRUM,
* Agile Unified Process,
* LEAN Development,
* Feature -Driven Development (FDD),

Dynamic System Development Methodology (DSDM),

* Microsoft Solution Framework (MSF).

Bu metodolojilerden en yaygınlarından birsi olan Scrum’ı ele alalım.

**Scrum Nedir?**

Scrum, çevik yazılım geliştirme durumunda uygulanabilecek bir proje yönetim metodolojisidir. Sadece yazılım geliştirmek için değil diğer türlerden projeler için de uygulanabilir. Doksanlı yılların ortasında oluşturulan bu metodoloji, karmaşık ve gereksinimlerinde öngörülemeyen değişimlerin yaşanabileceği projeleri sprint adı verilen küçük projelere böler. Böylelikle, projenin tümünü tek seferde planlayıp geliştirici ekibi yoğun ve yorucu tempoda çalışmaktan kurtarıp, değişen gereksinimlere karşı daha hazır kılar. Geliştirici ekip 5-9 kişiden oluşan takım/takımlarla birlikte Scrum Master yani ekip liderinden oluşur. Geliştirici ekip, günlük ayaküstü toplantılar yaparak bir çeşit raporlama işlemi yapar ve bu olay proje geliştirme aşamasında gelinen nokta hakkında bilgi verirken aynı zamanda karşılaşılan sorunları da ortaya çıkarır. Böylelikle projeyi geliştirme açısında daha sağlıklı bir süreç izlenir.

**Scrum Yapısındaki Temel Kavramlar:**

**- Roller (Roles)**

**\***Ürün sahibi (Product Owner): Proje sonucu ortaya çıkan ürünün sahibi kişi veya firma.

**\***Scrum Takımı (Scrum Team): Projeyi geliştirip ürünün ortaya çıkmasında rol alan ekip.

**\***Scrum lideri (Scrum Master): Takım ile iletişim halinde olup eksiklerini karşılayan ve buyurgan tavırlar sergilemekte olmayıp iletişimini sağlıklı bir şekilde yapan kişi.

**- Toplantılar (Meetings)**

**\***Günlük Scrum Toplantısı (Daily Scrum Meeting): Takım üyelerinin yaptığı iş ve yapacağı işlerin için rapor verdiği, ayaküstü 10-15 dakikalık rapor verme/alma işlemi. “Dün ne yaptım, bugün ne yapacağım ve beni engelleyen şeyler neler ?” gibi sorulara cevap alınır.

**\***Sprint Gözden Geçirme (Sptint Review): Tamamlanacak sprint için yapılması planlanan görevleri, ürün sahibi ile takım incelemeye alır.

\*Sprint Planlama (Sprint Planning): Gereksinimlerin detaylı olarak belirlenmesi, risk analizlerinin yapılması, dağıtım ekibinin belirlenmesi gibi planlamaların yapıldığı kısımdır.

**-** **Bileşenler/Araçlar (Tools)**

\*Ürün Gereksinim Dokümanı (Product Backlog): Projenin gereksinimlerini, müşteri de sürecin içerisinde rol alacak şekilde detaylı olarak analiz edilip öncelik sırasına göre kayıt altına alınmasıdır.

\*Sprint (Koşu) Dokümanı (Sprint Backlog): Her küçük proje için Product Backlogdaki önceliğe göre yapılmasının planladığı, bitirilmiş ve yapılmakta olan işlemlerin kayıt altına alındığı kısımdır.

\*Sprint Kalan Zaman Grafiği (Burndown Chart): Proje için önceden belirlenen bitirme süresiyle ne kadar sürede bitirildiğinin karşılaştırılmasını sağlayan grafiktir.

Çevik yazılım geliştirme metodolojileri, sağladığı büyük avantajlar ve modern dünyanın gerektirdiği ihtiyaçları karşılaması sebebiyle yazılım geliştirme ekiplerince oldukça sık kullanılmaktadır. Bu metodolojiler arasında yer alan Scrum ise GOOGLE, IKEA, BBC, MICROSOFT, YAHOO, IBM, NOKIA ve SIEMENS gibi büyük firmaların kullandığı popüler bir proje yönetim biçimidir.

**Scrum Neden Popüler?**

Scrum, proje ekibinin motivasyonunu yüksek tutması, maliyeti azaltması, projenin geliştirme aşamasında değişen ihtiyaçlara uygun olması ve büyük başarı oranı sunması nedeniyle bu metodoloji popülerdir ve milyonlarca ekip tarafından tercih edilir. Kısacası Scrum’ı Scrum kılan özellikler aynı zamanda popülerlik kazandırmıştır.

**Kaynakça:**

* Doç. Dr. Deniz Kılınç, Bakırçay Üniversitesi Yazılım Mühendisliğine Giriş Dersi 2. ve 3. Hafta Sunumları
* <https://clarusway.com/what-is-software-development-life-cycle/>
* <https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/>
* <https://toptalent.co/agile-yazilim-gelistirme-alaninda-bir-metod-scrum-framework-nedir>