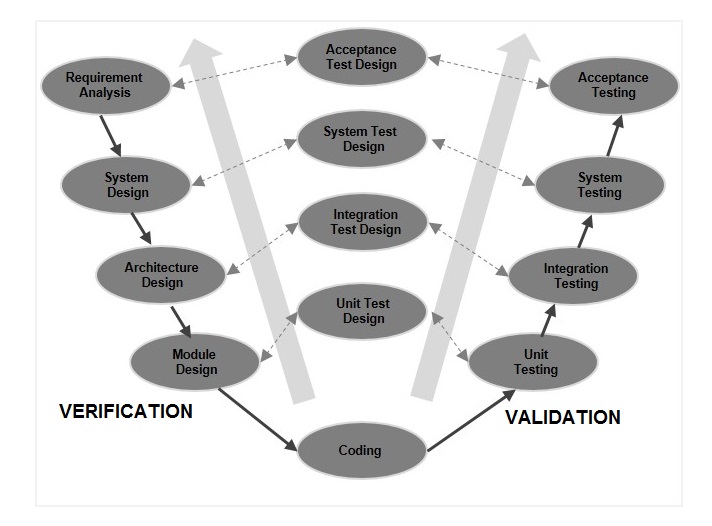
**YAZILIM YAŞAM DÖNGÜSÜ VE MODELLERİ**

Mert ÇİN

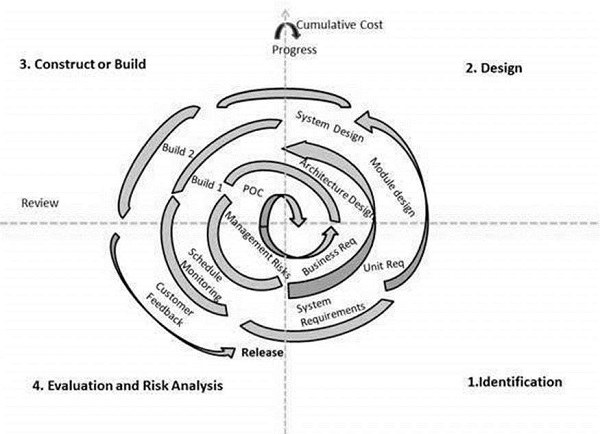
Yazılım yaşam döngüsü, yazılımın takip edeceği rotayı çizer. Yazılım üründür ve döngüden oluşan bir yaşam süreci vardır. Yazılım yaşam döngüsü temel basamaklara sahiptir: gereksinim, analiz, tasarım, gerçekleştirme, dağıtım ve bakım. Kendi içlerinde de maddelere ayırabileceğimiz bu basamakların her biri ayrı öneme sahiptir ve mutlaka takip edilmelidir. Yaşam döngülerinin amacı; her adımda kontrollü ve analiz ederek devam etme, maksimum verimlilik sağlama, işçi ve malzeme maliyetini düşürme, teslim hızını arttırma ve ihtiyacı karşılamaktır. Bununla beraber bilgi işlem gücü artarsa talep de artacağından basamaklara oldukça odaklanılmalıdır. Her basamak belirli sınırlar içinde daha fazla parçaya ayrılabilir. Yazılım yaşam döngüsü ilk temel basamağı olan gereksinim**;** proje hedeflerinin, şartlarının, sınırlarının ve isteklerin belirlendiği bölümdür. Amaç müşterinin gerçek ihtiyacını kavramaktır. Ön araştırma (plan) kısmından sonra ilk kapsamlı veri toplama ve karar verme kısmıdır. Kaynak ve maliye-fayda tanımları, kullanıcı ihtiyaçları mercek altına alınır. Tüm bunları oluşturmak için fizibilite çalışmaları tamamlanır. Fizibilite 5 parçaya ayrılabilir: operasyonel, finansal, teknik, insan faktörü, hukuki/siyasi. Sürecin devamı için gereksinim aşamasının iyi kurgulanması ve belgelenmesi gerekir. Bu adım başarılı bir şekilde tamamlandığında sıra analiz aşamasına gelir. Analiz, gereksinim aşamasında elde edilen verilerin ayrıştırılmasıdır. Analiz için sorunları küçük parçalara ayırmalıdır. Bir önceki basamağın iyileştirmelerini içerir. Tasarım altı alt dalda incelenebilir: mimari, kullanıcı arayüzü, platform, programlama, iletişim, güvenlik. Mimari; genel şablonun oluşturulduğu, program dilinin seçildiği bölümdür. Kullanıcı arayüzünde müşterinin yazılımla olan ilişkisi ele alınır ve yazılımın girdilere olan tepkisi kullanıcı bazında oluşturulur. Platform kısmında temelde iş sistemi kurulur. Programlamada sadece program diliyle ilişkili olmayız. Uygulamanın sorunlarını çözme, görevleri gerçekleştirme başarısını da kontrol etmeliyiz. İletişim programın diğer uygulamalarla olan iletişimidir. Güvenlik; SSL trafiği şifreleme, parola koruması, kimlik bilgisi saklama gibi konuları içerir. Bir uygulama içinde veri tabanı bulunduruyorsa bunun güvence altında olduğundan emin olunmalı ve müşteriler ve/ya kullanıcılar gözetilmelidir. Tasarım safhasının sonunda müşteriyi projeye dahil etmek, geri bildirim almak ve proje işleyişi hakkında temel fikir sahibi olabilmek adına prototip geliştirilebilir. Prototip, herhangi bir değişiklik veya iptal durumlarında henüz projenin başındayken bu kararı vermek için çok yararlıdır. Prototip, tasarım basamağıyla beraber açıklanabildiği gibi projedeki yerine veya kullanılan model gereğince ayrı bir basamak olarak da ele alınabilir. Her iki durumda da göz ardı edilemez bir basamaktır ve bazen maliyet açısından hayati olabilir. Tasarım kısmından sonra gerçekleştirme kısmı gelir. Gerçekleştirme basamağı en anlaşılır haliyle kod yazmaktır. Ürünün olgunlaştığı yerdir. Kod yazarken belirli kısımlarda vakit geçirmeden belgelemek önemlidir. Kodun anlaşılır olabilmesi için olabildiğince çaba harcanmalı ancak yine de yorum satırları kullanılabilir. Daha sonra kodun revizesi için yaptığımız tüm bu işlemleri kullanıcılar için de yapmak faydalı olacaktır. Kullanım kılavuzları, sık sorulan sorular veya videolu bir anlatım hazırlanabilir. Kodu tek kişi veya birden fazla kişi yazabilir. Belli bir büyüklükteki projelerde birden fazla kod yazarı vardır. Bu bağlamda tüm yazarlar iletişimi asla koparmamalı, beraber ilerlemelidir ve birbirlerinin kodlarını görebilmelidir. Birden fazla yazarın olduğu senaryoda, projede birçok görev vardır. Sorumlulukları parçalayarak devam edilmelidir. Bu görevlerden biri de test etmek ve hata ayıklamadır. Testlerin otomatikleşmesi için de bazı kodlar yazılabilir veya farklı ortamlarda yapılabilir. Test kısmı da aynı prototip gibi alt açıklamada yer alabilir. Ancak son zamanlarda artan işlev ve çalışan sayısıyla beraber test, paylaşılan bir görev değil uzmanlık alanı olmuştur. Bu nedenle güncel SDLC basamaklarında gösterilmektedir. Bir diğer kısım dağıtım, uygulamanın sunulduğu aşamadır. Bu bir linkin paylaşılması, mobil uygulama aplikasyonun markette sunulması olabilir. Güncelleme veya yeni bir versiyonun entegre edilmesi de bu kapsamdadır. İşletme ve bakım kısmında ise artık sahaya inilir. Testte karşılaşılmayan hatalar ortaya çıkar, yeni güncellemelerin konusu belirlenir. Bu temel basamaklardan biri olmamasına karşın konuya genel bir girişte olması gereken “Emeklilik” basamağı da açıklanmalı. Emeklilik, adından da anlaşılacağı gibi artık görevini tamamlamış, güncellemelerle yeterli düzeye ulaşamayacak veya altyapı değişikliğinin olduğu durumlarda yaşanır. Projenin bu kısmı tamamen emekliye ayrılır.

**SDLC MODELLERİ**

Geçmişten günümüze yazılım üretmek için projenin büyüklüğü, çalışanların yetkinliği, projenin türü vb. kriterlere göre farklı modeller kullanılmıştır. Tüm modellerde temel basamaklar benzerdir. Fark yaratan yürütülme biçimidir. Doğru modeli seçmek için gereksinimlere, ihtiyaçlara, isteklere hâkim olunmalıdır. Bir diğer faktör ise ortamdır. Modeller için tek bir doğru yoktur zaten tek bir doğru olsaydı farklı seçeneklerin varlığından bahsedemezdik. Popüler modellerin öncesinde Gelişigüzel model, Barok model veya Kodla-düzelt modeli gibi profesyonel olmayan modeller mevcuttur. Bu modeller genelde eski zamanlarda ve tek kişinin çalıştığı projelerde kullanılmıştır. Dokümantasyon ve planlama geri planda kalır. Başlıca modeller ise şelale (waterfall) modeli, V süreç model, spiral model, artırımsal model, prototipleme modeli ve çevik modellerdir.

**Şelale Modeli:** En eski modellerden biri olarak bilinen Şelale Modeli, Winston Royce tarafından yaratılmıştır. Gereksinim, analiz, tasarım, uygulama, bakım ve emeklilik basamaklarını barındırır. Bu basamaklar temel olarak böyle olmasına karşın -daha önce de konusu geçtiği gibi- kişi veya proje bazlı olarak değişebilir. Eski zamanların popüler modellerindendir. Çıktığı dönem için ayırt edici özellikleriyle fark yaratmış ve yeni bir bakış açısı getirmiştir. Zamanla değişen sektörel ihtiyaçlar ve müşteri istekleriyle dezavantajları ağır basmaya başlamıştır. Bu model öncesinde plansız ve sınırları çizilmemiş gereksinimler olduğundan birçok sorun ortaya çıkıyordu. Şelale modeli katı kuralları ve iyi belgeleme istemesiyle bu sorunu çözdü. Şelale avantajları bakımından basit, anlaşılır, geniş kaynak havuzunun bulunduğu, kullanımı ve yönetimi kolay, tek tek işlenen aşamalara sahip, kilometre taşları üstüne kurulu, iyi belgelenmesi gereken, net sınırları çizilen ve nispeten düşük maliyetli bir model olarak karşımıza çıkar. Öte yandan en önemli dezavantajı proje sürerken değişen gereksinimleri karşılayamamasıdır. Bu sebeple müşteri sadece ilk süreçte varlık gösterebilir. İyi belgelenmemiş bir projede entegre yapılması neredeyse imkansızdır. Sadece kısa süreli projelerde kullanılabilir. Uzun süreli, sürekliliği olan, karmaşık ve nesneye yönelik programlamalarda kullanılamaz. Şelale modelinin diğer özellikleri ise durağan ve doğrusal olmasıdır. Ancak bir basamak tamamlandığında diğerine geçilebilir ve bu sebeple çevik değildir. Yeni sürümlerin piyasayı sürülme sıklığı 1-2 yılı bulabilir.

**V Süreç Modeli:** Şelale modelinin bir miktar geliştirilmiş halidir ancak şelale modeline çok benzerdir. Yine net gereksinimlere ve çok disiplinli bir yapıya sahiptir. Bu modeli Şelale modelinden ayıran en önemli fark isminden de anlaşılacağı gibi V şeklinde ilerlemesidir. Türkçeye çevrildiğinde ikisi de “doğrulama” anlamına gelse de sol tarafı “Verification”, sağ tarafı “Validation” üzere iki kısmı vardır. En altında “Kodlama” kısmı bulunur. Verification, nesnel doğrulama; Validation, öznel doğrulamadır. Verification; iş gereksinim analizi, sistem tasarımı, mimari tasarım, modül tasarımından oluşur. İş gereksinim analizi, müşteri ihtiyaçlarını anlama kısmıdır. Müşteri, gereksinimler konusunda kararsız kalacağından bu kısmı iyi yönetmek gerekir. Müşteriyle ayrıntılı iletişim içerir. Bu kısımdan alınan girdilerle “Kabul Testi” kısmının planı yapılabilir. Sistem tasarımı, gereksinimler tamamlandığında ürünün donanım ve kurulumunun detaylandırıldığı yerdir. Buna dayanarak daha sonra zaman kazanmak için “Sistem Test” planı yapılabilir. Mimari tasarım, mimari özelliklerin anlaşılıp tasarlandığı yerdir. Çok fazla teknik yaklaşım içinden finansal ve teknik fizibiliteler sonucu karar verilir. Sistem tasarımından farklı işlevli modüller ayrılarak yüksek seviyeli tasarım (HLD) oluşur. Dâhili modüllerle iletişim kurulur. Entegrasyon planı tasarlanır ve belgelenir. Modül tasarımı, ayrıntılı dâhili tasarım yapılarak düşük seviyeli tasarım (LLD) oluşturulur. Diğer modüllerle uyumu önemlidir. Kontroller yapılarak tasarıma yön verilir. Kod kısmı programlama dilini belirleme, kodlama, test ve optimizasyon içerir. Validation; birim test, entegrasyon testi, sistem testi ve kabul testinden oluşur. Birim test, hazırlanmış kod üzerinden yürütülen testtir. Entegrasyon testi, mimari tasarımla ilgilidir. Dahili modüllerin uyumu ve iletişimi test edilir. Sistem testi, sistem tasarımıyla ilgilidir. İşlevselliği ve geliştirilmekte olunan sistemlerin uyumunu ölçer. Yazılım ve donanım dengesini ayarlar. Kabul testi, ilk kısım olan iş gereksinim analiziyle ilgilidir. Kullanıcı ortamında test edilir ve performans ölçülür.

**Spiral Model:** Barry Boehm tarafından 1986 tarihli "A Spiral Model of Software Development and Enhancement" makalesinde tanımlanmıştır. Spiraller halinde yinelemeli sürekli kontrollerle ilerleyen risk odaklı bir yazılım geliştirme modelidir. Kimlik, tasarım, inşa, risk analizi olmak üzere dört aşamadan oluşur ve bunu sürekli yineler. Kimlik, iş gereksinimiyle başlar. Spirallerde sırasıyla sistem gereksinimi, alt sistem gereksinimi ve birim gereksinimdir. Spiralin sonunda ürün pazara çıkar. Bu adım müşteri ve sistem analisti arasında köprü vazifesi görür. Tasarım, kavramsal tasarımla başlar ve sırasıyla mimari tasarım, modül-mantık tasarımı ve fiziksel ürün tasarımı olarak devam eder. Spiral sonunda son tasarım oluşmuş olur. Çevirisi yapıldığında benzer anlamlara gelen iki kavram karşımıza çıkmakta: “construct” ve “build”. Bu kısımda construct için ilk ürün fikri, geliştirme aşaması diyebiliriz. Kavram kanıtı (PoC: Proof of Concept) karşımıza çıkar. Bu kavramı “bir şeyin olabilirliğini kanıtlamak” olarak açıklayabiliriz. Müşteriden geri bildirim almak için kullanılır. Build için ise yapı kelimesini kullanmak daha açıklayıcı olacaktır. Fikrin daha net bir şekilde oluşmasıdır. Bu kısımda ürünün çalışan bir sürümü ortaya çıkar. Genel yapı belirlenir. Müşteri geri bildirimi beklenir. Risk analizi ise teknik fizibilite, zaman ve para maliyeti, tahminler ve izleme içerir. Risk analizi sonunda bir değerlendirme yapılır ve yine bir geri bildirim süreci söz konusudur. Her yinelemeye geri bildirimlere göre yön verilir. Bu modelde her spiral sonunda radikal değişiklikler beklenir. Değişen gereksinimleri karşılayabilen model, kullanıcı/müşteriyi ilk safhadan bu yana olaya dahil eder. Küçük parçalı yinelemeler halinde ilerler. Prototip bazlıdır. Her spiral bittiğinde bir prototip oluşur. Tüm bunlarla beraber çok karmaşık ve yönetimi zordur. Spiral iyi kontrol edilmezse gelişim sonsuz bir döngüye girer. Bunu çözmenin yolu ise detaylı dokümantasyondur. Ara aşamanın fazlaca bulunmasından dolayı fazlaca dokümantasyon gerektirir. Spiral model, bütçenin kısıtlı veya risksiz olması gerektiği projelerde kullanılabilir. Büyük projelere daha uygundur çünkü bu kadar karışık bir model küçük çaplı projeler için pahalı olabilir. Yinelemelerin sonu gelmemesi ihtimaline karşı zaman sınırı olan projelere uygun değildir. Bu yüzden uzun soluklu işlerde kontrolü elden bırakmamak için kullanılır.

**Artırımlı Model:** Birden fazla bağımsız modüle ayrılmış aşamalı tasarımlardır. Minik sistemlerin birleşimi olarak düşünülebilir. Hem geliştirme hem de bakım sürecini ilerletir. Her modülde gereksinim, tasarım, test ve uygulama bölümleri sabittir. Sonraki her sürüm bir öncekine bir işlev ekler ve eklenerek son ürüne ulaşmak amaçlanır. Bu modelin en önemli özelliklerinden biri hata ayıklaması kolaydır. Çok fazla alt sistemden oluştuğu için her şey apaçık gözlenebilir. Bu kadar fazla sistem bir plan içinde yönetilmelidir. İyi bir planlama, ayrıntılı bir takvim ve yüksek seviye gereksinim tablosu gerektirir. Esnek bir yapıdadır. Programcıların nispeten yetersiz olduğu bir projede kullanılabilir. Hızlı teslim için idealdir ancak maliyeti yüksek bir modeldir.

**Prototipleme Modeli:** Adı üstünde prototip üretme mantığında ilerler. Gereksinim, analiz, tasarım gibi hazırlık kısımları hızlıca tamamlanarak ürün üretimine başlanır. Üretilen prototip kullanıcılara sunularak geri bildirimlere göre geliştirmeler yapılır. Bu model hızlıca piyasada boy göstermeyi sağlasa da karmaşık ve kirli bir ilerleme içerir. Geri bildirimler dikkate alınmazsa performans düşer. İletişim bu modelde fazlasıyla yer alır. Tam oturmayan bir proje olduğundan fikir ve kod çalınması gibi durumlar yaşanabilir. Prototip olduğu bilinse de müşteri her zaman en güzel ürünü kullanmayı bekler. Çıkarılan ürün prototip haliyle yetersizse tam sürümün itibarını kötü etkileyebilir. Hızlı olmak olumsuz bir sonuç çıkarabilir.

**Çevik modeller:** Hızlı ve çok yönlü yazılım geliştirmeyi sağlar. Aşamalı olarak 1-4 hafta süren teslimatlar yapılarak dinamik bir süreç yürütülür. Daha öncesinde var olan bazı kavramlar değişmiştir. Süreçlerin ve araçların yerini bireyler ve etkileşimler almıştır. Ağır dokümantasyon angaryasından sürekli çalışan bir yazılıma geçilmiştir. Yine bir yığın kontrat hazırlamaları yerine müşteri ilişkileri tercih edilmektedir. Amaç, bir planı takip etmek yerine değişikliklere hızlı şekilde yanıt verebilmek olmuştur. Çevik modellerde öncelik erken ve sürekli teslimattır. Hem müşteriyi olabildiğince proje içinde etkin kılmak hem de geliştiricilerin ürünü sürekli somut biçimde gözlemlemesini sağlamak amaçtır. Ürün fikirleri için kullanılan bir yöntem MVP (Minimum Valiable Product)’dır. MVP, “Minimum uygulanabilir ürün” anlamına gelir. Eric RIES yapmış olduğu tanımda; ekibin, müşterilerin geri bildirimleri hakkında en az çabayla en fazla miktarda doğrulanmış öğrenme toplamasını sağlayan ürün olduğunu söyler. En az çabayla gerçek ya da gerçeğe yakın ürünü geri bildirim ya da -Eric RIES’in söylemiyle- doğrulanmış öğrenme toplamak için piyasaya sürmek MVP’nin temelidir. Bazı ürünler arka tarafta otomasyonu tamamlanmadan, tasarımı yapılmış haliyle bile müşteri karşısına çıkabilir. Ürün hakkında toplanmak istenen geri bildirime göre değişiklik gösterir. Ürün kullanılacak kadar kaliteli değildir. Minimum ifadesini gözden kaçırmamak hatta beklentileri daha da aza indirmekte fayda vardır. MVP, proje sahipleri için ürünün gidişatını görmeyi sağlar. Olası iptal veya revize kararlarında para ve zaman maliyetini ciddi ölçüde azaltır. Diğer modeller için maliyeti kesinlikle arttıracak ancak çevik modeller için de en ayırt edici özellik değişen gereksinimlere projenin sonunda bile yanıt verebilmektir. Bu da ancak dinamik yapının sürdürülebilir olmasıyla mümkün olur. Proje içindeki insanlar, şirketler, belgeler ve makinelerin birlikte, stabil bir şekilde korunması ve düzen içinde devam edilmesi gerekir. Proje sahipleri ve müşteriler projede beraber olmalıdır. Geliştiriciler bilgi alışverişini çok önemsemelidir. Düzenli aralıklarla işin durumunu takip etmek için toplantılar yapılmalıdır. Takım kendi kendinin motivasyon kaynağıdır. Dış etkiye ihtiyaç duymadan örgütlenmelidir. Teknik mükemmeliyet ve iyi tasarım çevik yazılım geliştirmeyi ilerletecek kavramlardır. Bu kadar çevik ve dinamik bir iş çıkarma periyodunda yapılması gereken önemli bir iş de gereksiz olanı ayıklamadır. Öncelik olmayan olan işlerden kaçınmak ve sadelikle iş yürütmek çevikliği mutlaka destekleyecektir. Bu konudaki sloganımı da sizlerin beğenisine sunuyorum: “Yeterli olanla yetin!”

Çevik yazılım geliştirme modelleri, yeni dünyanın tüm gerekliliklerini karşılamak için oluşturulmuştur. İnsanların değişen özelliklerine ayak uydurmaktadır. Sabırsızlık ve değişkenlik hızından yazılım modelleri de nasibini almıştır. Müşteriler veya kullanıcılar hemen ürünü görmek, bir istekle değişiklik yapmak ve zaman alan işlerden tümüyle sıyrılma eğilimindeler. Diğer yönden artık etrafta çok fazla uyaran olduğundan daha sade olmak, bir kesim için ciddi bir tercih meselesidir. Bir profesyonel yazılımda, tercih edilme için profesyonel olmayan bir yaklaşım olsa da bağımsız değildir. Bugün insanların yazılım kullanma sıklığı o kadar artmıştır ki artık profesyonellerin yanında son kullanıcı kitlesi de ciddi söz sahibidir. Bu bağlamda yazılımlarda minimalist yaklaşımlar kesinlikle ihtiyaç olmuştur. Kısaca çevik modeller sadece gelişmiş değil aynı zamanda yenilenen dünyanın zihniyetini yansıtan modellerdir.

Çevik modellerin alt başlıkları şu şekildedir: eXtreme Programming (XP), SCRUM, Agile Unified Process (AUP), Feature-Driven Development (FDD), Test-Driven Development (TDD), LEAN Development, Dynamic System Development Methodology (DSDM) ve Microsoft Solution Framework (MSF).

**SCRUM:** Bunların içinden en popüleri olarak nitelendirebileceğimiz proje yönetim metodudur. Karışık yönetim süreçlerinin idaresinde, parçalar halinde tekrarlı olarak kullanılır. Düzenli geri bildirime dayalı planlamaların yapıldığı bir kavramdır. Müşteri ihtiyacına yönelik ve iletişim ön plandadır. Çevik modelinin gerekliliklerini sağlar. Üç önemli başlığı mevcuttur. İlki şeffaflık, yapılan işlerin ve ilerlemenin herkes tarafından izlenebilmesidir. Diğeri denetleme, erişim hakkı verilen kişiler tarafından değişiklikler yapılabilir, birlikte ilerleme sağlanır. Sonuncusu ise uyarlama, gereksinimlerde yapılan bir değişiklik veya proje planında yapılan bir değişikliğin kolayca uygulanabilmesidir.

SCRUM konusunun kendine ait terimleri mevcuttur. Roller ikiye ayrılır: Pig roller ve Chicken roller. Pig roller aktif rol oynayan kişileri tanımlar. Chicken roller ise tam tersi aktif rol oynamayan kişilerdir. İlk olarak “Product Owner”, Geliştirme Takımı ile müşteri ilişkisini sağlar. Projeyi tanımlar ve product backlog’u hazırlar. Gerekli durumlarda sprintleri iptal etme hakkı bulunur. “SCRUM Master”, SCRUM kurallarına, teorisine ve pratiğine hâkim olan isimdir. Tüm bu uygulamalardan sorumludur. Yöneticilik yapmaz, düzeni sağlar. Ortamda herhangi bir rahatsızlık varsa onu çözer, verimi en üst seviyeye çıkarmayı amaçlar. “Geliştirme Takımı”, sprint içerisinde verilebilecek tüm işleri yapabilecek yetkinliğe sahip kişilerdir. Sprint backlog’u hazırlar. Sorumluluk verilmesine, yöneticiye ihtiyacı yoktur. Geliştirme işinden sorumludur. Belirli işleri devralmazlar, çapraz görevlendirmeyle iş bölümü yaparlar. Chicken rollere ise müşteri ve satıcı gibi kişiler örnek verilebilir. Kavramlar incelendiğinde rollerin tanımında da ismi geçen “product backlog”, gereksinim listesidir. Product owner tarafından hazırlanır. Dinamik bir biçimde yeni gereksinim ekleme veya çıkarma yapılabilir. “Product backlog item”, her bir gereksinime verilen isimdir. “Sprint”, görevin küçük kısımlarına verilen isimdir. Tüm aktiviteler öncelik sırasına koyularak sprintler içinde tamamlanır. 1-4 hafta arası sürer. “Sprint backlog”, product backlog listesindeki gereksinimleri önceliklerine göre sıralayarak sprint yerleştirilmesidir. Zaman çizelgesi oluşturularak planlanır. “SCRUM board”, sprintte tüm işleri “yapılacak görevler – görev sürüyor – teste hazır – tamamlandı” maddelerine ayırır ve buna göre takip eder. “Burndown chart” ise bu görevlerin grafiğidir. X ekseninde sprint günleri, Y ekseninde sprinti tamamlamak için kalan günleri gösterilir. Bu grafik şeffaflık maddesine hizmet eder. Son kategori ise toplantılardır. Bu kadar hızlı devam eden projelerde iş takibi yapmak çok önemlidir. Pig rollerin katıldığı Sprint Planlama toplantıları vardır. Yeni sprintler product owner’a sunulur. Herkes görevi kendisi üstlenir. Daily SCRUM; kısa, öz ve sürekli yapılan günlük toplantılardır. Genelde 15 dakika kadar ve ayaküstü yapılır. 24 saat planlanır. Dün ve bugünün işleri konuşulur. Bir sorun varsa SCRUM Master’a bildirilebilir. Kişiler beklenmez. Sadece çok eksik durumunda iptal olur. Sprint retrospective, sprint boyunca yapılan takip toplantılarıdır. Sprint review, sprint sonunda yapılan değerlendirme toplantısıdır. Yeni sprintin belirlenip sunulmasıyla toplantı döngüsü yeniden başlar.

**eXtreme Programming (XP):** İlk kez 1990’lı yılların ortalarında “Chrysler Comprehensive Compensation (C3)” programında kullanılmıştır. Daha sonra Kent BACK’in dâhil edilmesiyle ivme kazanmıştır. Bu ekipten çıkan birkaç kitap ile kullanıcılar bilgilendirildi. XP, çevik modellerin içinde en spesifik olanı olarak tanımlanır. Çevik olmanın özelliklerini taşır. Dinamik değişen gereksinimlere uygun, küçük veya orta çaplı ekiplere sahip, birim ve işlevsel otomatikleştirilmiş teknolojiye sahiptir. Döngüler halinde bir haftalık ve üç aylık olarak iş takibi yapılır. XP için iletişim, basitlik, geri bildirim, saygı ve cesaret temel yapı taşları denebilir. Bakım ve revize kolaylığı için basit ve sade tasarımlar üretilir. Bir diğer öne çıkan kavram ise cesarettir: Geri bildirimlere göre hareket edebilmek bazen revizelere veya tamamen ürünü çöpe atmaya kadar gidebilir. Bunu yapmak için cesaret gerekir. Bunun yanında tüm süreçler devam ederken saygı da korunmalıdır çünkü XP takım çalışmasını öne çıkarır. XP içinde öne çıkan birkaç farklı yöntem vardır. Bunlardan ilki iki kişilik kadrolardır. Tek bilgisayarda iki kişi kod yazmak işlemidir. Kod verimini arttırmak amaçlanır. Hatalar erkenden tespit edilir ve zamandan kazanç sağlanır. Diğeri ise önce test yöntemidir. Normal süreçte kod-test-değerlendirme iken bu yöntemde başarısız test kodu yazılır, çözümü araştırılır. Daha sonra buna göre kod yazılır ve hata en aza indirilebilir. İncelenen son yöntem artırımlı tasarımdır. Projenin genişliğini anlamak için ön çalışma yapılır. Sonradan spesifik bir alana odaklanılır. Ayrıntıları incelenir. Bu yöntemde değişiklik maliyetlerini azaltmak ve güncel tasarım kararları alabilmek amaçlanır. XP modelinde SCRUM’daki gibi belirgin roller yoktur. Herkes geliştirici olarak projede yer alır. Ancak geliştiriciler arasından bir müşteri rolü, bir gözcü rolü, bir de koç rolü belirlenir. Müşteri çevik modellerde çok önemli bir yere sahip olduğundan dâhil edilecek müşteri, iş kurallarını sorgular. İşi takip edecek bir gözcü ve XP için diğerlerinden tecrübeli bir koç bulunur.

Linkedin: linkedin.com/in/mert-çin-96bb05230

Medium: [medium.com/@mertcin0](mailto:medium.com/@mertcin0)

Github: [github.com/iammertcinceng](https://github.com/iammertcinceng)

**KAYNAKÇA**

JEVTIC Goran, What is SDLC? Phases of Software Development, Models, & Best Practices, <https://phoenixnap.com/blog/software-development-life-cycle>, 15 Mayıs 2019

PETERSON Julie, Software Development Life Cycle: Finding a Model That Works, <https://www.whitesourcesoftware.com/resources/blog/sdlc-software-development-life-cycle/>, 25 Haziran 2020

ÇOR Seçil, SCRUM NEDİR?, <https://medium.com/@secilcor/scrum-nedi%CC%87r-6a4326951dd8>, 27 Mart 2018

tr.wikipedia.org

en.wikipedia.org

tutorialspoint.com

javatpoint.com

agilealliance.org