

Yazılım Süreçleri

- Yazılım Süreçleri, yazılım üretimi için birbirleriyle uyumlu etkinlikler dizisidir.
- Yazılım krizi neticesinde bir metodoloji olması gerekliliği.
(Geleneksel ve Agile süreç modelleri)

Geleneksel Yazılım Süreç Modelleri

- Gelişigüzel model
- Barok Model
- Çağlayan (Şelale-Waterfall) Model
- Artırımlı Geliştirme Modeli
- Evrimsel Model
- Rational Birleşik Süreci (RBS) Modeli
- Döngüsel (Spiral) Model

- Farklı yazılım sistemleri sebebiyle evrensel bir yazılım mühendisliği yöntemi bulunmamaktadır.
- Farklı yazılım süreçleri olsa da ortaya konulacak yazılım, yazılım mühendisliğinin temel süreçlerini içermelidir.

- Gereksinim mühendisliği sürecinde ana etkinlikler;
 - Gereksinimlerin çıkarılması ve analizi
 - Gereksinimlerin spesifikasyonu ile belgeye dönüştürülmesi
 - Gereksinimlerin geçerlenmesi ile test edilmesi

Süreç Etkinlikleri

- Yazılım sisteminin spesifikasyonu, tasarımı, gerçekleştirimi ve sınanması süreçleridir.
- Bu süreç etkinlikleri çağlayan modelde art arda iken, artırılmış modelde birbiri içine geçmiş durumdadır.
- Gereksinin yönetim sistemleri, tasarım model editörleri, program editörleri, test araçları ve hata ayıklayıcılar gibi çeşitli araçlardan faydalanılabilmektedir.

Yazılım Çözümleme / Spesifikasyonu

- Yazılımın fonksiyonelliği ve uygulanmasındaki kısıtlamalar tanımlanır.
- Gerçeklenecek sistemi anlamaya yönelik çalışmalardan ve üst düzey planlama eylemlerinden oluşur (Uygulama alanı, Kullanıcı gereksinimleri, program parçaları arasındaki ilişkiler vb.).

- Sistemin sunması gereken servislerin anlaşılması ve tanımlanması kısıtların belirlenmesi sürecidir.
- Bu aşamadaki yanlış kararlar büyük sorunlara yol açabilmektedir.
- Gereksinim mühendisi yazılım için fizibilite çalışması yapar.

- Çevik yöntemlerde gereksinim spesifikasyonu ayrı bir etkinlik değil, sistem geliştirmenin bir parçasıdır.
- Gereksinimler gayriresmi olarak kullanıcı önceliklerine göre belirlenir.

Yazılım Tasarım

- Yazılımı geliştirme sürecinin çeşitli dönemlerinde izlenecek yol ve işlemler tasarlanır.
- Yazılım tasarımına yönelik şemalar (UML şemalar), devre şemaları, mimari vb.

- Çevik geliştirme yaklaşımında tasarım ve gerçekleştirim iç içe geçmiş durumdadır ve biçimsel tasarım belgesi üretilmez.
- Yazılım tasarımı; yazılımın yapısı, veri modeli ve yapılar, arayüzler ve algoritmaların tanımıdır. Tasarım aşamalar halinde geliştirilir ve geliştikçe geri dönüşlere göre değişiklikler yapılır.
- Model güdümlü yaklaşımda tasarım çıktıları tasarım diyagramlarıdır. Çevik geliştirme yöntemlerinde tasarım çıktıları ayrı spesifikasyonu belgeleri yerine programın kodunda temsil edilebilmektedir.

Yazılım Geliştirme (Gerçekleme)

- Spesifikasyonları karşılayan tasarım bir programlama dili ile üretilir.

Yazılım Geçerleme (Sinama)

- Müşterinin isteklerinin yerine gelip gelmediği kontrol edilir.
- Erken teşhis ile ileride oluşabilecek hataların önüne geçilebilir.
- Hataların çoğu tasarım aşamasında gerçekleştiğinden erken müdahale edebilmek için sık sık test edilmeli.

- Sistemler tek parça olarak sınanmamalıdır.
- Bileşen testi (Sistemi oluşturan bileşenler), sistem testi (bileşenler arasında beklenmeyen ve bileşen-arayüz sorunlarından kaynaklanan hatalar) ve müşteri testi tek tek test edilmeli ve ardından entegre sistem test edilmelidir.

- Artırımlı geliştirmede, her artırım geliştirildikçe gereksinimlerine göre sinama yapılmalıdır.
- Çevik süreçlerde test güdümlü geliştirme ile test, geliştirme başlamadan gereksinimlerle paralel geliştirilir. Böylece zamandan tasarruf sağlanır.
- Plan-güdümlü yazılım sürecinde sinama, bir dizi test planı ile gerçekleştirilir.

Yazılım Evrimi (Bakım)

- Yazılım zamanla değişen müşteri ihtiyaçlarına göre evrim geçirmelidir.
- Yazılımdaki tasarım, kodlama, gereksinim hataları düzeltilmelidir.
- Yazılım farklı bir platforma taşınması halinde gereklidir.

Yazılım Süreç Modelleri

- Modeller yazılım geliştirme sürecinin yapısını ve adımlarını belirler.
- Geliştirilecek projeye göre model seçilmelidir.
- Her süreç modeli, süreci belli bir açıdan ele alır. Böylece süreçle ilgili kısmi bilgi vardır.

Gelişigüzel Model

- Gelişigüzel :)
- Tekrarlanabilir başarılar için sıkıntılı bir model.

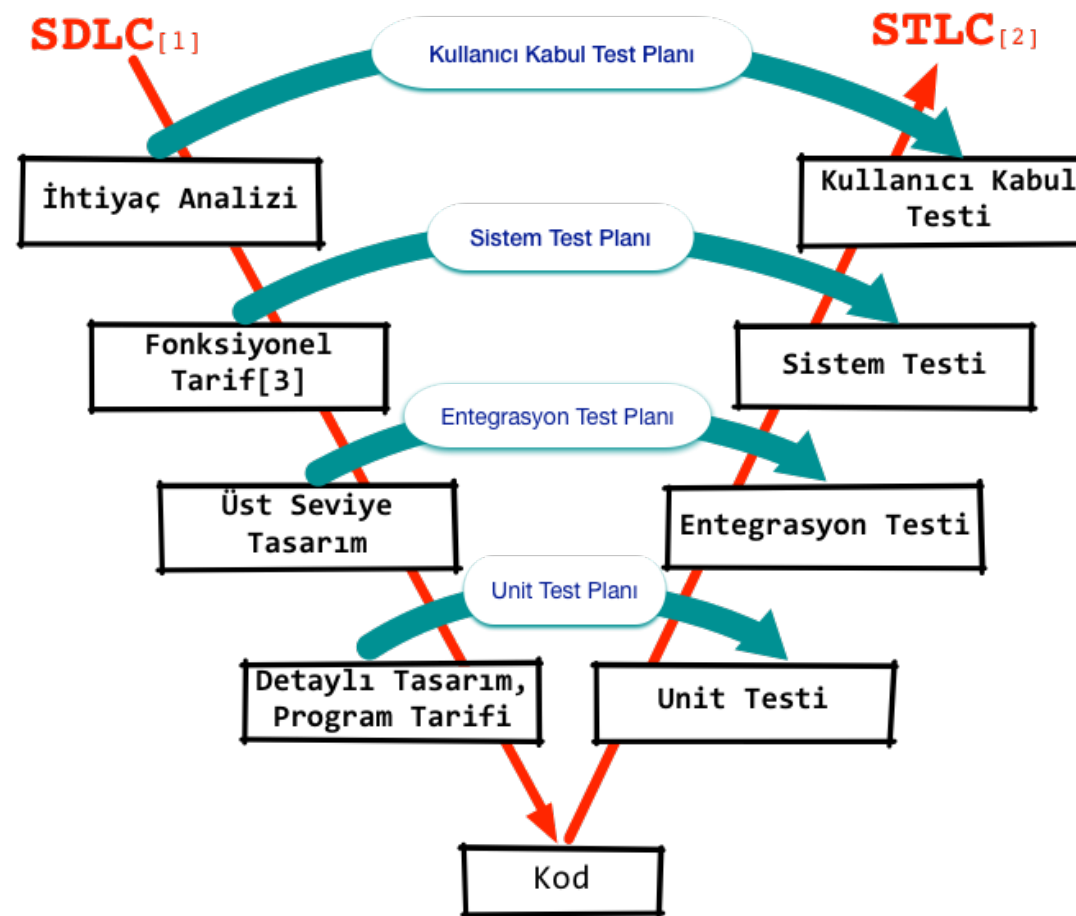
Barok Model

- Çekirdek süreçler doğrusal bir şekilde gerçekleştirilir.
- Terminoloji farklı olsa da yazılım süreçlerini içerir.
- Belgelemeye önem veren bir modeldir.
- Aşamalar arasında geri dönüş gerekliyse nasıl yapılacağı belirsizdir.
- Mevcutta kullandığımız süreç tanımlama yöntem ve metodolojilerinde belgeleme için ayrı bir işlev bulunmamaktadır. Belgeleme süreç içinde yapılmaktadır. Ayrı bir aşama değildir.

Şelale (Çağlayan, Ardışıl) Model

- Çözümleme ==> Tasarım ==> Kodlama ==> Sinama ==> Bakım.
- Adımlar sırayla ilerler ve bir hata tespit edilirse bir önceki adıma geri dönülür. Ancak ileriki adımlarda meydana gelen hataların düzeltilmesi yada revize edilmesi zahmetlidir.
- En eski ve askeri modellerden yola çıkarak oluşturulmuş modeldir.
- Süreç uzun sürebilir.
- Her safha sonunda dokümantasyon yazılmalıdır. Eğer safha sonunda dokümantasyon ve test olmadıysa safha tamamlanmamış sayılır.
- Dokümantasyon ayrı bir süreç olarak ele alınmaz.
- Müşteri sistemi en son görür.
- Kısıtlı sistem türleri için kullanılabilir.
- Emniyet-Kritik yazılımlar için oldukça uygun bir modeldir.
- Müşteri gereksinimlerini tam anlatamayabilir.

V Süreç Modeli



- Sol taraf üretimi, sağ taraf sınımayı gösterir.
- Belirsizliklerin az, iş tanımlamalarının belirgin olduğu bilgi teknolojileri projeleri için uygun bir modeldir.
- Kullanıcı yazılım projesinin geliştirilmesi sürecinde projenin içindedir.

Evrimsel (Prototip Geliştirme) Model

- Müşteriyle birlikte çalışılarak sistem zaman içinde gelişir.
- Çok birimli organizasyonlarda kullanılmak için geliştirilecek projeler için önerilen bir modeldir.
- Sistem zaman içinde kazanılan anlayışa ve müşteriden gelen geri dönüşlere göre sürümler şeklinde geliştirilir.
- Öğrenme amacıyla müşteri için prototipler oluşturulur.
- En iyi bilinen gereksinimle başlanarak az bilinene doğru geliştirilmeye devam edilir.
- Modelin başarısı geçirdiği ilk evrimin başarısına bağlıdır.
- Bu süreç modelinde müşteri gereksinimlerini anlamak kolaylaşır. Böylece proje geliştirme sürecinde ortaya çıkacak riskler ve hatalar azalır.
- Sistem bakımını yapmak zordur.

- Kavramların gösterilmesi, tasarım seçeneklerinin değerlendirilmesi ve sorunlar ile olası çözümleri hakkında bilgi edinilmesini sağlamak için geliştirilen erken versiyondur.
- Sistem tasarım sürecinde, yazılım çözümlerinin araştırılıp kullanıcı arayüzü geliştirmede etkilidir.
- Yazılımın güçlü ve zayıf yanları ortaya çıkar.
- Tasarımın fizibilitesinin kontrolü için kullanılabilir.
- İlk örnekleme maliyetini azaltmak ve zamandan tasarruf için bazı işlevler göz ardı edilebilir. (hata işleme ve yönetimi, güvenilirlik ve program kalitesi vb.)

Artırmalı Geliştirme Modeli

- Çözümleme, Gerçekleme ve Sınama süreçlerini dönüşümlü olarak ele alır.
- Çağlayan ve evrimsel süreç modellerini temel almaktadır.
- Sistem her versiyonun bir öncekine fonksiyonellik kattığı artırımlar şeklinde geliştirilir.
- Başlangıç gerçekleştirimi, kullanıcı ve diğer kişilerden geri bildirim alma ve gereken sistem geliştirilene kadar yazılım evrimine dayanır.
- En yaygın kullanılan modeldir.
- Müşteri gereksinimleri önemlerine ve birbirine olan bağlılıklarına göre önceliklendirilir.
- Yazılım geliştirilirken yapılan değişiklikler hem maddi açıdan hem de kolay olması açısından önemlidir.
- Nesneye yönelik programlama metodolojisi ile uyumludur.

- Çağlayan modeline göre avantajları;
 - Gereksinim değişikliği maliyeti düşüktür. Tekrar yapılması gereken analiz ve belge miktarı azdır.
 - Müşteriden geri bildirim almak daha kolaydır.
 - Yazılımın müşteriye erken teslimi mümkündür.
 - Bütün sistem hazır olduğunda eğitim sürecine ihtiyaç duyulmaz.
 - Aynı anda kullanım ve üretim yapılabilir.
 - Öncelikli gereksinimler erken teslim edildiği için daha çok test edilir.
 - Erken teslimler sonraki teslimler için prototip görevi görür ve gereksinimlerin ortaya çıkarılmasında yardımcı olur.

- Dezavantajları;
 - Süreç görünür değildir. Sistem hızla geliştikçe her versiyonu yansıtan belgelerin üretilmesi maliyetlidir.
 - Yeni artırımlar eklendikçe sistemde yapısal bozulmalar meydana gelebilir.
 - Büyük ekiplerde iş iyice sıkıntılı hale gelebilmektedir.
 - Her artırım müşteriye teslim edilmek zorunda değildir.
 - Yeni yazılımın denenmesi süreci normal iş süreçlerini aksatabilir.
 - Yeni sistem başka bir sistemin yerine geçecek ise süreçte zorluklar meydana gelir.
 - Artırım gerçekleştirilene kadar gereksinimler tam tanımlanmadığı için sıkıntılar meydana gelebilir.
 - Spesifikasyon yazılım ile beraber geliştirildiğinden bu yapı bazı kurumların sözleşme şartlarına uymaz.

Bütünleştirme ve Konfigürasyon Modeli

- Mevcut yazılımın yeniden kullanılmasına odaklanan geliştirme süreci, yeniden kullanılabilen yazılım bileşenleri ve bu bileşenlerin bileşimi için bütünleştirme çerçevesine dayanır.
- Maliyet ve risk azalır.
- Zamandan avantaj sağlar.
- Oluşturulan sistem kullanıcıların tüm ihtiyaçlarını karşılayamayabilir.
- Sistem evrimi üzerindeki kontrolün bir kısmı yitirilebilmektedir.

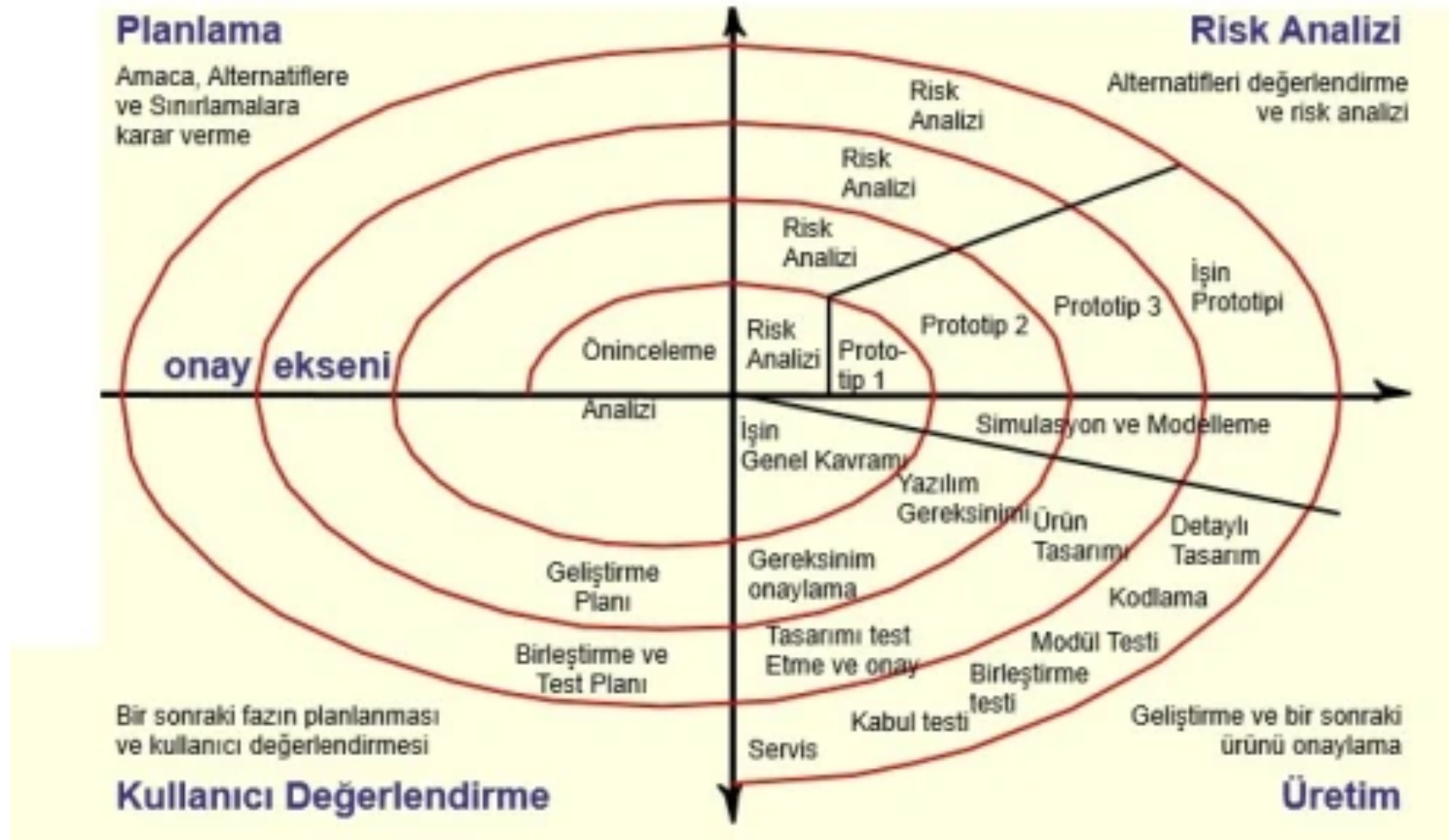
- Şu üç tür yeniden kullanılabilmektedir;
 - Tek başına kullanılabilen uygulama sistemleri. Belirli bir uygulamada kullanılabilmesi için adaptasyonu gereklidir.
 - Java Spring benzeri bir bileşen ile entegre edilerek, paket olarak geliştirilenler.
 - Servis standartlarına göre geliştirilen ve internet üzerinden kullanılan web servisleri.

- Aşamaları;
 - Gereksinimlerin spesifikasyonu: sistemin ilk gereksinimleri önerilir.
 - Yazılım keşif ve değerlendirme: Gereksinimler için gereken fonksiyonelliği sağlayacak bileşen ve sistemler için arama yapılır.
 - Gereksinimleri düzenleme: Gereksinimler yeniden kullanılabilir bileşen ve uygulamalar hakkındaki bilgiler kullanılarak düzenlenir.
 - Uygulama sistemi konfigürasyonu: Gereksinimleri karşılayan kullanıma hazır bir sistem varsa konfigüre edilerek sistemde kullanılabilir.
 - Bileşen adaptasyonu ve bütünleştirme: Kullanıma hazır bir sistem olmaması durumunda yeniden kullanılabilir bileşenlerin değiştirilmesi yoluyla yeni bileşenler geliştirilir.

Döngüsel (Spiral / Helezonik) Model

- 4 aşamadan oluşmaktadır;
 - Planlama
 - Risk değerlendirme ve azaltma
 - Geliştirme ve doğrulama
 - Planlama
- Risk odaklı bir yazılım sürecidir.
- Yinelemeli artımsal bir yaklaşım sunar.
- Riskler süreç boyunca ele alınıp çözümlenir ve minimuma iner.
- Her bir aşamada 4 alt aşama gerçekleştiği için her adımda tekrarlanabilir başarı elde edilir.
- Üretim süreci boyunca prototipler geliştirilir ve bunlar kullanıcı tarafından sınanır.
- Kodlama ve sinama daha erken başlar.

Helezonik(Spiral) Modeli



- Değişen sistem gereksinimleriyle baş edebilmek için;
 - Prototip geliştirme
 - Artırımlı teslimat

Rational Birleşik Süreci (RBS) Modeli

- Genel modellerden evrensel bir süreç modeli geliştirmek için ortaya çıkan bir modeldir.
- Diğer modellere benzer süreçler oluşturma amaçlı farklı yollarla somutlaştırılabilen esnek bir modeldir.
- IBM tarafından kullanılsa da çok yaygın kullanımı yoktur.

Kaynakça

- Ian Sommerville - Yazılım Mühendisliğine Giriş
-
- <https://www.mobilhanem.com/yazilim-gereksinim-analizi/>
- Seker
- Gül, Z. (2006). Yazılım geliştirme sürecinin iyileştirilmesi ve Türkiye uygulamaları, İstanbul Teknik Üniversitesi, (Doctoral dissertation, Fen Bilimleri Enstitüsü)
-
- <https://systemanalyze.wordpress.com/gelistirme-metotolari/>
- Dr. Öğr. Üyesi Yunus Emre SELÇUK'un ders notları