**İSA ER 210601708**

**Bakırçay Üniversitesi Bilgisayar Mühendisliği 1. Sınıf**

**YAZILIM – YAŞAM DÖNGÜ MODELLERİ**

1. **MODELLERİN AÇIKLANMASI**
   1. **Çağlayan-Şelale Döngü Modeli**

Şelale, yazılım geliştirme sürecinin yayınlanan ilk modelidir.1970 yılında Royce tarafından sistem mühendisliği sürecinden türetilmiştir. Model bir genellikle olarak adlandırılan geliştirme yazılımının aktivite merkezli klasik modeli geleneksel model. Şelale modeli sıralı bir SLCM'dir. Aşamalar boyunca sürekli olarak aşağı doğru aktığı görülmektedir. Gereksinim analizi, tasarım, uygulama, test etme (doğrulama), entegrasyon ve bakım.

Bu model, aşağıdaki gibi çıktılarla aşamaların her biri boyunca yolda ilerler: her aşamada yazılım gereksinimleri spesifikasyonu, tasarım belgeleri, kod vb.

Her aşama, yapılması gereken belirli bir dizi faaliyet ve çıktıdan oluşur. Sonraki aşama başlamadan önce tamamlanır.

Bu modeldeki kilit nokta, bir aktivite tamamlandıktan sonra asla geri dönmemektir. Yani her aktiviteyi ve aşamayı bir gözden geçirme ile takip etmek esastır. Kısacası, model varsayar yazılım geliştirme sürecinin adım adım bir süreç olarak planlanabilmesi kullanıcı ihtiyaçlarını koda dönüştürür

* 1. **V Model**

V-Shaped veya V Model, şelale modelinin sıralı bir yolu olan bir varyasyonudur. Süreçlerin yürütülmesi. Her aşama bir sonraki aşamadan önce tamamlanmalıdır. Bu modelde şelale modelinden daha çok test etme üzerinde durulmuştur. Bu test prosedürleri, herhangi bir kodlama yapılmadan önce yaşam döngüsünün başlarında geliştirilir, uygulamaya devam eden aşamaların her biri sırasında. Gereksinimler hayata başlar. Döngü modeli tıpkı şelale modeli gibi. Geliştirmeye başlamadan önce, bir sistem deneme planı oluşturulur. Test planı, aşağıda belirtilen işlevselliği karşılamaya odaklanır. Gereksinimlerin toplanması. Üst düzey tasarım aşaması, sistem mimarisine odaklanır ve tasarım. Parçaları test etmek için bu aşamada bir entegrasyon test planı oluşturulur. Yazılım sistemlerinin birlikte çalışabilme yeteneğidir. Ancak, düşük seviyeli tasarım aşaması, gerçek yazılım bileşenlerinin tasarlandığı ve birim testlerinin gerçekleştirildiği yerdir ve bu aşamada oluşturulmuştur. Uygulama aşaması, yine, tüm kodlamanın yer alır. Kodlama tamamlandığında, yürütme yolu sağdan devam eder. V’nin daha önce geliştirilen test planlarının şimdi kullanılmaya başlandığı tarafı.

* 1. **Helezonik – Spiral Modeli**

Spiral Model, 1988 yılında Barry Boehm tarafından geliştirilmiştir. Spiral Modelde geliştirme çabası yinelemelidir ve bir yineleme tamamlanır tamamlanmaz başka bir iterasyon başlar. Model, faaliyet merkezli bir

model olup şelale modelindeki zayıflıkların kaynağı. Temel amacı, yazılım geliştirme sırasında sık olmayan değişiklikler. Modelin riski var aynı faaliyetlere ek olarak yönetim, yeniden kullanım ve prototipleme faaliyetleri şelale modeli vardır. Genişletilmiş faaliyetler döngüler ve turlar halinde yapılır. Her biri yuvarlak şelale modelini takip eder ve hedeflerin belirlenmesini, belirtilmesini içerir. kısıtlamalar, alternatifler üretme, riskleri tanımlama ve çözme, geliştirme ve sonraki seviye ürün ve planlama aktivitelerinin doğrulanması Spiral modelin bir başka özelliği de, sürecin sadece bir döngüsünün aslında yazılım çıktıları geliştirin. Spiralin merkezinden başlayarak, her geliştirme aşamasının (işletim kavramı, yazılım gereksinimleri, ürün tasarımı, ayrıntılı tasarım ve uygulama) sarmalın bir döngüsünü içerir.

Model, yüksek riskli geliştirme projeleri ile uğraşırken ve bir gerçek zamanlı uygulamalar gibi gereksinimlerden tam olarak emin olmayan müşteri. Bu model çoğunlukla büyük devlet projeleri için kullanılmaktadır

* 1. **Artımsal Geliştirme Modeli**

Artımlı Model, yinelemeleri olan şelale modelinin bir varyasyonudur. Model, gereksinimler kademeli olarak bölümlere ayrılabildiğinde kullanılabilir. bağımsız olarak geliştirilen bir dizi ürün. Başlangıçta, proje küçük parçalara bölünür. Bu, geliştirme ekibinin sonuçları göstermesine olanak tanır sürecin başlarında ve sistem kullanıcılarından değerli geri bildirimler alın. Çoğu zaman, her yineleme aslında bir aşamadan gelen geri bildirimle mini bir Şelale sürecidir. sonraki aşamanın tasarımı için hayati bilgiler sağlar. Bunun bir varyasyonunda model, her adımın (veya bir dizi sürecin sonunda üretilen yazılım ürünleri adımlar) artımlı sürümler olarak hemen üretime geçebilir Yazılı kullanım yoluyla projenin ömrü boyunca orta düzeyde kontrol sağlanır. dokümantasyon, kullanıcı ve teknoloji yönetimi tarafından resmi inceleme ve onay belirlenen önemli kilometre taşlarında. Paydaşlara somut kanıtlar sunulabilir yaşam döngüsü boyunca proje durumu. Model, gereksinimlerin başlangıç ​​aşamasında iyi bilindiği ve ürün, yapı artışları adı verilen bağımsız çıktılara bölünebilir. İletişim ve koordinasyon becerileri proje geliştirmenin merkezinde yer alır. Ayrıca, tasarımında kazanılan bilgi olarak bilgi paylaşımını sağlar. ilk artış, ikinci artış tasarımına aktarılabilir.

* 1. **Evrimsel Prototipleme Modeli**

Değerlendirmeli Prototipleme modeli, bir başlangıç ​​geliştirme fikrine dayanmaktadır. Uygulama, bunu kullanıcı yorumuna sunun ve birçok versiyon aracılığıyla rafine edin yeterli sistem geliştirilene kadar. Model açıkça genişletiyor gereksinimler aşamasına artımlı model. İlk yapı artışı için kullanılır ikinci bir yapı artışı için gereksinimleri iyileştirin. Kullanıcılara yapılan ilk artış, yayınlandı ve bu, geliştirilmesine yardımcı olacak geri bildirim sağlar. Sonraki artışlar için gereksinimler. Ayrıca, bir yapı artışı geliştirmek gerçekten işe başlamadan önce tanınmayan sorunlara görünürlük sağlamak bu artış üzerine. Gereksinimler anlaşıldıktan sonra tasarımın aşamaları, kodlama, artımlı geliştirme içinde şelale modeli ile uygulanabili modeli. Bu modelin ana avantajı, riski erken dönemde ele alma yeteneğine sahip olmasıdır. proje, nihai sistemin kabul edilip edilmeyeceği ve görünür olup olmayacağı konusunda erken geri bildirim proje boyunca ilerleme. Bu modelin kullanılmasında önemli olan faktörleri deneyimli geliştiriciler kullanmak, program ve bütçe beklentilerini yönetmek ve prototipleme etkinliğinin kendisinin yönetilmesi. Muhtemelen geliştiricilerin sıklıkla sahip olabileceği iş sistemleri için en uygun olanıdır. son kullanıcılarla gayri resmi etkileşimler. Bu, gereksinimler değiştiğinde yararlıdır hızlı bir şekilde, müşteri bir dizi şartı yerine getirmekte isteksiz olduğunda veya kimse uygulama alanını tam olarak anlamıyor. Ancak, bazı riskler ortaya çıkacaktır. modeli kullanarak. Modelle ilgili ana riskler gerçekçi olmayan program ve bütçe beklentileri, prototiplemenin verimsiz kullanımı, gerçekçi olmayan sistem performansı ve kötü tasarım. Modelin kullanılmasıyla karşılaşılacak sorunlar şu şekilde sınıflandırılabilir: yönetim, bakım ve sözleşme. Mevcut yönetim süreçleri, bir şelale modeli geliştirme ve uzmanlık becerileri gerekli olabilir ki bunlar tüm geliştirme ekiplerinde mevcut olan yönetim sorunlarıdır. Üstelik, sürekli değişim sistem yapısını bozma eğilimindedir, bu nedenle uzun vadeli bakım pahalı.[6]

* 1. **Birleşik Döngü Modeli**

Birleşik model, Boehm'in Spiral Modeline benzer başka bir yaşam döngüsü modelidir. bir proje, her biri bir ürünün teslimi ile biten birkaç döngüden oluşur. Her döngü, başlangıç, detaylandırma, inşaat ve geçiş. Yine her aşama bir dizi yinelemeden oluşur. Başlangıç ​​aşamasında bir fikir tanımlanır ve uygulanabilirliği değerlendirilir. İçinde detaylandırma aşaması, proje planlanır, sistem tanımlanır ve kaynaklar tahsis edildi. İnşaat aşaması geliştirme sürecine karşılık gelirken geçiş aşaması, kurulum ve geliştirme sonrası sürece karşılık gelir. Ayrıca, Birleşik Model gereksinimlerin, analizin, tasarımın, Modelin yüksek izlenebilirliği, değişikliklerin etkisinin anlaşılmasını sağlar. Kullanma bileşen tabanlı mimari, kolayca genişletilebilen bir sistem oluşturur, yazılımın yeniden kullanımı ve sezgisel olarak anlaşılabilir. Model teknik olarak daha az izin verir sorunu daha iyi anlayabilecek yetkin bireyler daha büyük girdi Ayrıca, kullanım senaryolarını ve senaryoları kullanarak gereksinimleri yönetmek, hem işlevsel gereksinimleri yakalamada hem de yardımcı olmada çok etkili olduğu bulunmuştur. sistemin beklenen davranışlarını gözlemlemek. Yinelemeli ve artımlı tasarım, proje risk profilini azaltmaya yardımcı olur, daha fazla müşteri geri bildirimi sağlar ve yardım sağlar geliştiriciler odaklanmaya devam ediyor uygulama ve test etme, vurgulayan bu yinelemelerin her birine katılır. Kaynakların evrelenmesi, yazılım geliştirmenin bir yönü olarak ele alınmayan diğer SLCM'ler.[7]

**2. MODELLERİN KARŞILAŞTIRILMASI**

**2.1**

Çağlayan Modelinin Bazı Avantajları Şunlardır:

• Proje ilerlemesinin daha doğru bir şekilde izlenmesini, erken tespit edilmesini sağlar.

• Projeler daha yönetilebilir ve maliyet aşımı olmadan zamanında teslim edilir

• Bütçe ve çabayı tahmin etme kolaylığı

• Her aşamanın sonundaki incelemeler, kullanıcının katılımını sağlar

• Sistemi test etmek ve sürdürmek için kullanılan belgeleri oluşturur

• Boşa harcanan çabayı en aza indirerek kaynakları koruyun

• Teknik olarak zayıf veya deneyimsiz personel için iyi sonuç verir.

Bu modelin dezavantajları İse Şunlardır:

• Müşteriler, gereksinimleri eksiksiz, doğru ve net biçimde vermeyebilir.

• Planlama ve belgelendirme için çok fazla zaman harcanması.

• Proje sonunda entegrasyon ve test için yoğun bir çaba gerekmektedir.

• Proje bitiminden önce herhangi bir tanıtım mevcut değildir.

• Gereksinimlerdeki değişiklikler ve hataları gidermek için yedekleme yapmak zordur ve

Pahalıdır.

• Esneklik eksikliği. • Tüm ihtiyaçları önceden tahmin etmek zordur.[2]

**2.2**

Bu modelin avantajları şunlardır.

• Basit ve kullanımı kolay.

• Her aşamanın belirli çıktıları vardır.

• Erken geçiş nedeniyle şelale modeline göre daha yüksek başarı şansı yaşam döngüsü boyunca test planlarının geliştirilmesi.

• Gereksinimlerin kolayca anlaşıldığı küçük projelerde işe yarar.

Bu modelin dezavantajları:

• Şelale modeli gibi çok sert

• Az esneklik ve kapsamı ayarlamak zor ve pahalıdır.

• Yazılım, uygulama aşamasında geliştirilir, bu nedenle erken yazılımın prototipleri üretilir.

**•** Test aşamalarında bulunan sorunlar için net bir yol sağlamaz.[3]

**2.3**

Avantajları aşağıdaki gibi sıralanabilir:

• Seçenek yelpazesi, mevcut modellerin iyi özelliklerini barındırır

• Risk odaklı yaklaşım, birçok zorluğun üstesinden gelir

• Yaşam döngüsü evrimi, büyümesi ve değişimleri için hazırlığı barındırır. yazılım ürünü

• Yazılım kalite hedeflerini yazılım ürünü geliştirmeye dahil eder

• Hataları ve çekici olmayan alternatifleri erkenden ortadan kaldırmaya odaklanır. Bununla birlikte, modelin aşağıda verilen birkaç zorluğu ve dezavantajı vardır

• Belirli son tarihlerin belirlenmemesi, aşağıdaki gibi şelale modeliyle sonuçlanabilir:

• Esneklik ve özgürlük, hesap verebilirliğin ve kontrolün kaybedilmesine neden olabilir. sözleşme yazılımı

• Tutarlılık, izleme ve kontrol sağlanabilir.[4]

**2.4**

Artımlı modelin avantajları şunlardır:

• İlk teslimatı yapmak için daha az maliyet ve zaman gerekir

• Daha hızlı sonuçlar sağlayın, daha az ön bilgi gerektirir ve daha fazlasını sunar

esneklik

• Daha küçük sistem geliştirme, daha az risk sağlar

• Artımlı finansmana izin verilir

• Müşteri tüm aşamaları içerir ve hızlı bir şekilde uygulanır

Dezavantajları:

• Parçalar, aşağıdaki durumlarda hizmetten çekilebilir, üzerinde yeniden çalışılabilir ve yeniden yayınlanabilir:

gereksinimler kararlı veya eksiksiz değil.

• Zor uygulama sorunları gecikti.

• Bazı modüller diğerlerinden çok daha erken tamamlanacak.

• İyi tanımlanmış arayüzler gereklidir.

• Her aşamayı takip eden kullanıcı geri bildirimi, müşteri taleplerinin artmasına neden olabilir.[5]

**3.HANGİ PROJEDE HANGİ MODELİ KULLANMALIYIZ?**

Artımlı, evrimsel ve spiral yazılım geliştirme modelleri yinelemeli geliştirme modellerindendir. Artımlı geliştirme modelinde ihtiyaçlar başlangıçta büyük oranda bellidir. Modelde sistem fonksiyonlara ayrılmaktadır. Yani müşteri ürünü kullanmak için tüm sistemin bitmesini beklemez ve belirlenen kritik ihtiyaçlarla geliştirilen yazılım kullanılmaya başlanır. Hazırlanan prototiplerle müşteri deneyim kazanır ve süreç içerisinde müşteriye daha çok yer verilir. Şelale modeline göre daha az risklidir ve herhangi bir başarısızlık tüm projeye yayılmaz. En başta kritik ihtiyaçlar belirlendiği için sistemdeki önemli parçalar en çok teste tabi tutulur ve bu parçalarda hata ile karşılaşılma olasılığı oldukça düşüktür.

Doğrusal modeller şelale ve V modeli olmak üzere iki sınıfa ayrılmaktadır. Klasik çevrim modeli olarak da adlandırılan şelale modeli genellikle gereksinimleri tam olarak belirlenmiş ve gereksinimleri değişmeyen projelerde kullanılmaktadır. Şelale modeli analiz, tasarım, kodlama, test ve destek adımlarından oluşmaktadır. Şelale modelinde ilk olarak gereksinimler belirlenerek gözden geçirilir ve onaylama işlemi yapılır. Daha sonraki adım da yine gözden geçirilerek onaylanır. Sonraki adımlarda da bu şekilde “gözden geçir-uygunsa onayla” işlemi devam etmektedir. Genel olarak şelale modelinde bir adım kendisinden önceki adım tamamlanmadan başlamamaktadır. Örneğin; kodlama adımı, kendisinden önce yer alan tasarım adımı tamamlanmadan başlayamaz. Kodlama adımına geçildiği an tasarım adımının tamamının bittiği anlamına gelmektedir. Şelale modelinin odaklandığı nokta her adımın dokümanlarla gerçekleştirilmesidir. Bu model dokümantasyon odaklı olduğu için hem çok maliyetli hem de uzun zaman almaktadır. Maliyet nedeniyle son aşamaya bırakılan problemler müşteri ihtiyaçlarına dönüt vermeyi zorlaştırmaktadır (Gül, 2006). Müşteri ihtiyaçlarının net olarak anlaşıldığı ve büyük projelerde kullanılması önerilmektedir. Şelale modelinin en büyük dezavantajı ise değişikliklere karşı esnek olmamasıdır. Şelale modeli, yazılım geliştirme sürecinde “tasarımdan önce tanımla, kodlamadan önce tasarla” ilkelerini desteklemektedir (Munassar N.M.A., Govardhan A., 2012). V modeli, şelale modelinin biraz daha gelişmiş sürümü olarak adlandırılabilir. Kontrol adımları daha iyi organize edilmiştir. Modelin diyagram gösterimi V harfine benzediği için bu adı almıştır. V modeli test planlarının yaşam döngüsünün daha erken safhasında geliştirilmesi nedeniyle şelale modelinden daha başarılı bir model olarak görülmektedir.[9]

**4.SCRUM GÜNÜMÜZDE NEDEN POPÜLER?**

Son yıllarda yaygın olarak kullanılan ve çevikliği uygulayan Scrum Çatısı, sadece yazılım geliştirme amacıyla değil, aynı zamanda “donanım, gömülü yazılım, etkileşim fonksiyonları ağları, otonom araçlar, okullar, hükümet, pazarlama, organizasyonların işleyişini yönetmek ve günlük yaşantımızda birey ve toplum olarak kullandığımız hemen her şeyi” geliştirmek için kullanılıyor.Scrum Rehberi’ndeki Scrum tanımlaması ile hızlı bir giriş yapalım: “İnsanların mümkün olan en yüksek değere sahip ürünleri üretken ve yaratıcı bir şekilde geliştirirken, karmaşık ve adaptasyona açık sorunları ele alabildikleri bir çerçeve”.Bu tanım gayet kapsayıcı olmakla birlikte, büyük resimdeki pozisyonel durumu tam kavrayabilmek için ilk akla gelen Agile (Çeviklik) ve Scrum ilişkisi. Scrum, bir Agile uygulamasıdır, fakat tıpkı Agile gibi kendi de bir çatıdır. Süreç yönetim çatısı olan Scrum’da belirlenmiş kurallar, roller, etkinlikler ve eserler (artifacts) bulunmaktadır. Tüm bunlara uyum zorunludur. Aksi halde uygulanan Scrum olmaz, hatta yanlış uyarlamalar için ScrumBut (Scrum, Ama.) denmektedir. Scrum basit, anlaması kolay ve uygulaması zor bir çatıdır. 1990’ların başından beri kullanılmakta ve hatrı sayılır seviyede gün geçtikçe popüleritesi artmaktadır.

Scrum’daki kurallar kümesi bir bakışta oldukça mekanik ve sanki bir metodolojiymiş gibi adım adım uyulması gereken bir liste hissi yaratsa da, teoriye baktığımız zaman bu sadece basit bir çerçevedir/çatıdır. Bu çerçeve içerisinde aslında istenen deneysel davranıştır. Scrum’ın temelinde deneycilik (empiricism) vardır. Sürekli kontrol sağlanabilmesi, özellikle de risklerin kontrol altında tutulabilmesi için artımlı (incremental) bir yaklaşım kullanılır. Bu artımı sağlayan iterasyonlara sprint denir. Scrum teorisini destekleyen üç temel ayak: şeffaflık, gözlem ve adaptasyondur. Şeffaflık adından da anlaşılacağı üzere ortak bir dil kullanımı ve hatta anlayışı, görünürlüğü esas almaktır, gözlem ve adaptasyon içinse Scrum etkinlikleri zorunlu kılınır. Bu etkinlikler: Günlük Scrum, Sprint Planlama, Sprint Değerlendirme ve Sprint Retrospektifi’dir.

Scrum’ı Scrum yapan Scrum Teorisi’nin üç temel ayağı olan şeffaflık, gözlem ve adaptasyonun vücud bulmasını ve takım içerisinde güven ortamını yaratmasını Scrum değerleri üzerinden sağlamak esastır. içselleştirilmesi gereken beş temel değer: taahhüt, cesaret, odak, açıklık ve saygıdır. Bu değerler sağlanıp, korunduğunda gözle görülür bir ekip ruhuyla birlikte başarılı bir scrum uygulaması sağlanmış olur. Birbirine karşı açık, saygılı davranan ve ortak bir hedefe tek bir canlı organizmasıymışçasına odaklanabilen, verilen sözlerin/taahhütlerin adeta yazılı bir anlaşma gibi önemsendiği ve bu zeminde deneme cesaretini içinde barından, kaizen kültürünü benimseyip kendini geliştirebilen bir Scrum Takımı; benimsenmiş Scrum Teorisi ve değerlerinin olası sonucudur.

Scrum’ın zorunu kıldığı ve Sprint (iterasyon) içerisinde gerçekleşen dört etkinlik bulunmaktadır. Tüm bu etkinlikler gözlem ve adaptasyon için resmi birer fırsat niteliğindedir ve muhakkak belirli bir zaman kısıtına sahiptir. Scrum takımının potansiyel olarak hayata geçirilebilir şekilde ve takım tarafından belirlenmiş ‘bitti’ tanımına uygun olarak ürettiği artımla sonuçlanan Sprint, Scrum’ın kalbi olarak ifade edilmektedir.

Sprint, Sprint Planlama etkinliği ile başlar, planlama süresi 1 aylık bir sprint için 8 saatle kısıtlandırılmıştır. Etkinlik, ne ve nasıl olmak üzere iki soruya cevap arar. Takım tarafından ürün için ne yapılacağı belirlendikten sonra nasıl yapılacağının planlanmasıyla etkinlik sona erer. Sprint’in amacı belirlenmiştir. Ürün İş Listesi (Product Backlog), Sprint İş Listesi (Sprint Backlog), Artım/Ürün Parçası (Increment) olmak üzere üç Scrum Eser’i (Scrum Artifacts) bulunmaktadır. Ürün İş Listesi, ürünle ilgili ihtiyaç duyulan her şeyin listelendiği, herkes tarafından görülebilen, şeffaf ve Ürün Sahibi tarafından yönetilen bir listedir. İşlerin önem ve değerine göre sıralanmıştır. Aynı zamanda işle ilgili tahmini ağırlıklar da bu liste üzerinden görülebilir. Bu tahminler Geliştirme Takımı tarafından yapılır, ağırlıkların belirlenmesinde son söz her zaman geliştirme takımınındır. Üst sıralarda yer alan işler genellikle daha detaylıdır. Bu detaylar geliştirme takımının kapasitesinin %10’ununu aşmayacak şekilde düzenlenmesi gereken Ürün İş Listesi İyileştirme (Product Backlog Refinement) etkinliği sayesinde daha da anlaşılır hale gelir.

Sprint Planlama Etkinliği’nde en üst sıradan başlayarak işler incelenir, tahminler yapılır ve sprint süre zarfında hangi işlerin yapılacağı Ürün Sahibi ile anlaşarak Geliştirme Takımı tarafından seçilir ve yapılması taahhüt edilen bu görece küçük listeye de Sprint İş Listesi denir.

Bitti tanımı tüm Scrum Takımı için tek bir anlam ifade etmelidir, ortak dil kullanımı ortaya çıkacak ürün parçacığını planlarken bir rehber niteliğindedir. Bir iş için bitti dendiği anda, Scrum Takımı’na dahil olan herkes aynı şeyi anlamalıdır. Farklı Scrum Takım’larının birbirinden farklı ‘bitti’ tanımı olabilir. Tanım takım tarafından belirlenir. Takım için asgari standardı ifade eder. Herkes için ortak bir referans noktası oluşmasını sağlar. Scrum Takımı olgunlaştıkça ‘bitti’ tanımının gelişmesi beklenir. Daha yüksek kalite standartlarının dahil edildiği bir ‘bitti’ tanımı her Scrum Takım’ı için elbette ortak bir hedeftir.[8]

**KAYNAKÇA:**

1)<http://dspace.trakya.edu.tr/xmlui/handle/trakya/440>

2,3,4,5,6,7) <https://polen.itu.edu.tr/handle/11527/5761>

8) <https://listelist.com/scrum-framework-scrum-catisi/>

9) <https://dergipark.org.tr/en/pub/asbider/issue/41362/500096>