T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SCRUM YAZILIM GELİŞTİRME METODOLOJİSİ İÇİN YÖNETİM SİSTEMİ TASARIMI VE GERÇEKLENMESİ

VOLKAN BAYTAM

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN
PROF. DR. OYA KALIPSIZ

İSTANBUL, 2011

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

SCRUM YAZILIM GELİŞTİRME METODOLOJİSİ İÇİN YÖNETİM SİSTEMİ TASARIMI VE GERÇEKLENMESİ

Volkan BAYTAM tarafından hazırlanan tez çalışması 19.10.2011 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Prof. Dr. Oya KALIPSIZ

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Prof. Dr. Oya KALIPSIZ

Yıldız Teknik Üniversitesi _____

Prof. Dr. A. Coşkun SÖNMEZ

Yıldız Teknik Üniversitesi _____

Yrd. Doç. Dr. Feza BUZLUCA

İstanbul Teknik Üniversitesi _____

ÖNSÖZ

Bu tez çalışmasında; yazılım sektöründe şirketlerin yaşadığı sıkıntıları içeriden biri olarak analiz etmeye ve çözüm önerileri sunmaya çalıştım. Bu çalışma beni, yazılım sektöründeki sorunun temel nedenlerinden birinin kullanılan proje yönetim süreçlerinin işin doğasına uygun olmadığı ve değişiklik yapılması gerektiği fikrine götürdü. Çalışmada, öncelikle mevcuttaki klasik yöntemlere alternatif olarak geliştirilen ve projeleri başarıya götürmeye aday çevik süreçlerin en ünlü üyelerinden biri olan Scrum, yapılan literatür çalışmaları ile anlatılmaya çalışıldı. Çalışma esnasında benim gibi sektörün içinde olan bireyler ile fikir alışverişinde bulunularak kişilerin sürece bakış açıları alındı. Daha sonra temel olarak süreç seçiminin tek başına yeterli olmayacağı, aynı zamanda sürecin işletimi ve yönetiminin başarıda temel etken olduğu fikri ile süreci doğru yönetebilmek için kullanılabilecek bir yazılım sistemi geliştirdim. Ortaya çıkan ürünü tecrübe eden kişilerden alınan geri beslemeler ile özellikle süreçte yeni organizasyonlar olmak üzere tecrübeli organizasyonların da kullanabileceği ve başarıya katkısı olacak bir çalışmanın ortaya çıkması ve bu geri dönüşün çalışmayı gören insanlardan alınması bu işin beni mutlu eden meyvesi oldu.

Tez çalışması boyunca gösterdiğim çabada, tez hocam sayın Prof. Dr. Oya KALIPSIZ'ın her aşamada büyük yardımı oldu. Kendisine desteklerinden dolayı teşekkür ediyorum. Tez çalışmam sırasında beni yüreklendiren, destekleyen mesai arkadaşlarıma ve yöneticilerime teşekkür ederim. Ayrıca tezin yazılmasında ve uygulamanın geliştirilmesinde fikirlerinden yararlandığım meslektaşlarıma da yardımlarından dolayı teşekkür ederim.

Son olarak, hayatımın her döneminde olduğu gibi bu zorlu dönemimde de bana inanan ve beni destekleyen aileme sonsuz teşekkür ederim.

Ağustos, 2011

Volkan BAYTAM

İÇİNDEKİLER

Sayfa

KISALTMA LİSTESİ

FDD	Feature Driven Development
IIS	Internet Information Services
O/RM	Object-Relational Mapping
ROI	Return on Investment
RUP	Rational Unified Process
UP	Unified Process
XML	Extensible Markup Language
XP	Extreme Programming

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

SCRUM YAZILIM GELİŞTİRME METODOLOJİSİ İÇİN YÖNETİM SİSTEMİ TASARIMI VE GERÇEKLENMESİ

Volkan BAYTAM

Bilgisayar Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Tez Danışmanı: Prof. Dr. Oya KALIPSIZ

Günümüz yazılım geliştirme ortamlarında sık yaşanılan sorunlardan birisi de ürün geliştirme yaşam döngüsü içerisinde müşteri gereksinimlerinin sürekli değişmesidir. Klasik yazılım geliştirme süreçleri yapıları ve çalışma şekilleri nedeniyle bu değişime ayak uyduramamakta, projeler genellikle başarısızlıkla sonuçlanmaktadır.

Bu nedenle klasik süreçlere göre yeni bir felsefe ve modernlik getiren çevik süreçlerin yazılım projelerinde kullanımı her geçen gün artmakta, proje takımları bu süreçlere geçiş yapmaktadır.

Bu tez çalışmasında öncelikle çevik süreçlerin en çok kullanılan üyelerinden olan Scrum modeli incelenecektir. Sürecin tarihsel gelişimi, içeriği, kuralları yapılan literatür araştırmaları ile sunulacaktır.

Sonraki kısımda tez çalışmasının ana fikri olan süreç yönetim sistemi tasarımı ve gerçeklenmesi aşaması anlatılacaktır. ScrumMApp uygulaması detayları ile anlatılıp, sonuçları yorumlanacaktır.

Anahtar Kelimeler: Yazılım Geliştirme, Süreç Modelleri, Çevik Süreçler, Scrum, Yazılım Proje Yönetimi

ABSTRACT

DESIGN AND IMPLEMENTATITON A MANAGEMENT SYSTEM FOR SCRUM SOFTWARE DEVELOPMENT METHODOLOGY

Volkan BAYTAM

Department of Computer Engineering

MSc. Thesis

Advisor: Prof. Dr. Oya KALIPSIZ

One of the frequently experienced problems in today's software development environment is the changing customer requirements faced through the product development life cycle. Classic software development processes cannot adopt to these changes because of their structures and the way they advance, as a result, the projects often result in failure.

Therefore, the usage of agile processes which brings a new philosophy and modernity compared to the classical processes is increasing rapidly in software projects, and project teams migrate to agile development processes.

In this thesis, primarily, Scrum which is the most adapted member of the agile processes models will be examined. The historical development of the process, its content and rules will be presented with related literature research.

In the next section, the design and implementation of the process management system, which is the main idea of thesis will be explained. ScrumMApp application will be explained in details and the results will be interpreted.

Key words: Software Development, Process Models, Agile Process, Scrum, Software Project Management

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Yazılım projelerinde hedefin (müşteri memnuniyeti, kaliteli ürün...) net belirlenmiş olmasına rağmen, günümüzde gerçekleştirilen birçok projenin başarıyla tamamlanamadığı ya da oluşturulan yazılım sistemlerinin müşteri gereksinimlerini sağlayamadığı görülmektedir.

Bunun en büyük nedeni ise kullanılan yöntemlerin gereksinimlere cevap verecek yetenekte olmamasıdır. Diğer bir neden ise yazılım projelerinin ve süreçlerin takibinin, işletiminin ve yönetiminin başarılı olarak gerçekleştirilememesidir.

1.2 Tezin Amacı

Bu tez çalışması yukarıda belirtilen tespitler üzerine bu sorunlara çözüm getirmeyi amaçlamıştır.

Yazılım proje yaşam döngüsü içinde bulunan bireylere başarıya götürmeyi amaçlayan bir sürecin anlatılıp bu sürecin daha iyi yönetilebilmesi amacıyla bir araç geliştirmesi hedeflenmiştir.

1.3 Orjinal Katkı

Geliştirilen yazılım sistemi hem bu sürece yeni hem de bu süreçte tecrübeli organizasyonlara, süreç yönetimlerinde sürece özel bir yönetim sistemi ile başarılı bir şekilde, tüm kuralları ve orjinalliğiyle Scrum sürecini işletme olanağı sağlamıştır.

BÖLÜM 2

YAZILIM GELİŞTİRMEDE GELENEKSEL VE ÇEVİK SÜREÇLER

Yazılım, Türk Dil Kurumu tarafından “Bir bilgisayarda donanıma hayat veren ve bilgi işlemde kullanılan programlar, yordamlar, programlama dilleri ve belgelemelerin tümü ” olarak tanımlanmaktadır.

Günümüzde sektörel anlamda ise “Müşteri” olarak adlandırdığımız kişiler veya organizasyonların gereksinimlerini karşılama adına çeşitli yöntemlerle belli kısıtlar dahilinde (maliyet, zaman, kaynak) geliştirilen ürünlerdir.

Yazılım projelerinin ana hedefi sabit bütçe ve belirli bir zaman diliminde, müşteri gereksinimlerini tam olarak sağlayan, hatalardan arındırılmış ve müşterinin piyasadaki rekabet etme yeteneğini kuvvetlendirecek kaliteli bir yazılım sistemi geliştirmektir. Yazılım geliştirme süreçleri bu hedefe ulaşmak için kullanılan metodları ihtiva eder.

Yazılım geliştirmek, yeni bir ürün yaratmaya benzer ve yazılım projeleri süreç içerisinde değişerek şekillenir. Projelerde karşılaşılacak olan birçok sorun, yazılımın doğası gereği başlangıçta öngörülebilir olmaktan uzaktır. Özellikle son yıllarda iş hayatında rekabetin sertleşmesi ve rakiplerden farklılaşma ihtiyacı, müşteri gereksinimlerinde hızlı değişime yol açmaktadır. Bu yüzden, yazılım geliştirirken alışılagemiş klasik seri üretim yöntemleri birçok koşulda başarılı çözümler üretmekten uzaktır ve başarı için daha esnek yöntemlere ihtiyaç duyulmaktadır [1].

Yazılım geliştirme her ne kadar yeni bir ürün üretimi olsa da seri üretime göre farklılıklar taşır. Yüzeysel bir açıdan, seri üretim yaklaşımının ve yeni bir ürün geliştirmenin doğalarını Çizelge 2.1’de karşılaştıracak olursak [1]:

Çizelge 2. 1 Seri üretim ile yeni ürün geliştirme durumlarının karşılaştırılması

	Seri üretim yaklaşımı	**Yeni ürün geliştirmek**
Gereksinimler	Proje başlangıcında belirli	Değişken
Değişiklik	Kısıtlı	En üst seviyede

Anlaşılacağı üzere, seri üretim yaklaşımı proje başlangıcında detaylı bir analiz sonucu gereksinimlerin belirlenmesi, sonrasında da bu gereksinimler temel alınarak projenin sonlandırılmasına yönelik çalışır. Ancak, yeni bir ürün geliştirirken (yazılım) detaylı bir analiz ile birlikte tüm gereksinimlerin proje başlangıcında belirlenmesi pek mümkün değildir. Yeni bir ürün geliştirirken istekler ve bunun doğrultusunda gereksinimler sürekli değişim gösterme eğilimindedir [1].

Gerçek dünya koşullarında yazılım projelerine bakacak olursak [1];

- Proje başlangıcında müşteriler, tam olarak ne istediklerinden yüzde yüz emin değildirler.
- Müşteriler isteklerini net bir şekilde dile getirmekte zorlanırlar.
- Müşterilerin asıl istekleriyle ilgili detaylar proje başında fark edilebilir değildir, süreç içerisinde yüzeye çıkarlar.
- Müşteriler isteklerinin gerçekleştiğini gördükçe, proje gözle görülür sonuçlar üretmeye başladıkça, isteklerinde değişiklik yapma eğilimindedirler.
- Dış koşullar değişim içerisindedir ve projelerin bunlardan etkilenerek değişime uğramaları kaçınılmazdır.

İşte bu yüzden, yazılım geliştirmek müşteri odaklı olmayı ve değişikliklere uyum sağlayabilmeyi gerektirir. Bunun aksine klasik yöntemlerle ele alınan yazılım projeleri değişime uyum sağlamakta güçlük çekecek ve istenen başarıyı elde etmekte zorlanacaktır [1].

3.1 Yazılım Geliştirme Süreci

Yazılım geliştirme süreci sıkıntılı ve uzun bir dönemdir. Günümüz bilgi teknolojileri proje yönetimi uygulayıcıları yüksek kaliteli yazılımı belirli maliyet ve zaman kısıtlarında teslim edebilme baskısı altındadır. Bu nedenle yazılım projelerinin doğru hedefte gidebilmesi için doğru yönetilebilmeleri önem arz etmektedir [2].

Ancak yazılım geliştirme süreci genellikle yönetim aksaklıkları, yanlış yöntemler nedeniyle başarısızlıkla sonuçlanmakta ve süreç içerisindeki bireyler için sancılı geçmektedir.

Yazılımlar karmaşık çevre şartları içerisinde geliştirilirler. Karmaşıklık hem geliştirme ortamından hem de hedef ortamdan kaynaklanır [3].

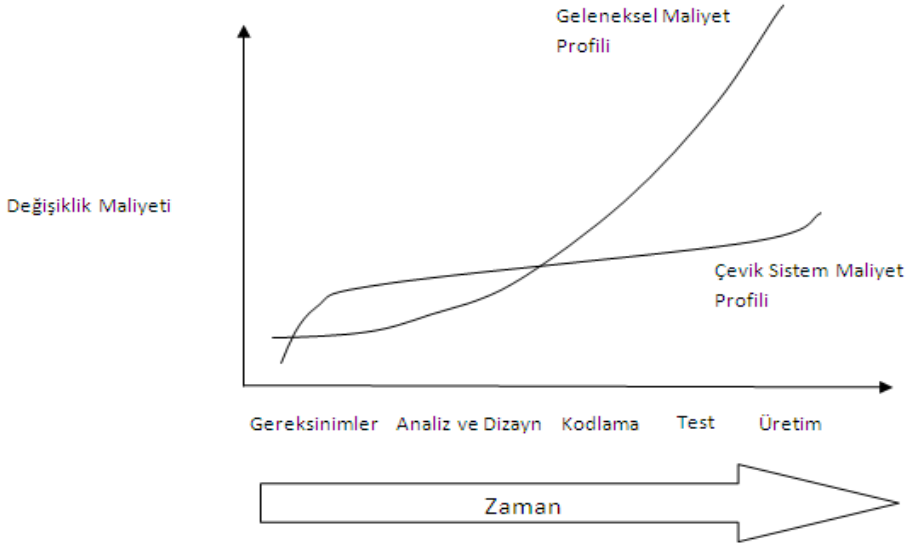
Karmaşıklığa neden olan çevresel değişkenler;

- Bireylerin yetkinlikleri: Teknolojideki, araçlardaki, metodlardaki yenilikler ve bireylerin bu konudaki bilgi birikimi
- Teknoloji adaptasyonu: Teknolojideki yenilikler sonucu oluşan düşük istikrar, bu yeniliklerin diğer teknoloji ve yöntemlerle olan uyarlama ihtiyacı
- Araçların gücü ve devamlılığı: Yeni ve daha güçlü geliştirme araçları bu araca yetkin uzman sayısında azalmaya neden olur ve bu araç işlevlerini daha kararsız yapıp, fayda yerine zarar getirebilir
- Metodların verimliliği: Hangi model, test, versiyon kontrolü ve tasarım metodları kullanılırsa kullanılsın buna bağlı olarak etkinlik, verimlilik değişir
- Ekip uzmanlığı: İş ve teknoloji anlamında yetkin uzmanlardan oluşan ekipler ile bu konuda yetkinlik kısıtı olan ekipler farklılık gösterir
- Yeni özellikler: Ürüne hangi yeni özellik eklenirse eklensin mevcut işlevsellikle ne derece uyumlu olacağı, süreç için risk teşkil edip etmeme durumu
- Metodoloji: Geliştirme yapılırken kullanılacak yöntemlerin katı veya esnek oluşu, bireylerin veya kurumun geleneklerinin yatkınlığı
- Rekabet: Proje esnasında gerçekleştirilecek mücadeleler. Yeni fonksiyonalitelerin duyurulması veya dağıtılmasının rekabete etkisi

- Zaman, Kaynak: Kullanılabilecek zaman ve geliştirme maliyeti
- Diğer değişkenler: Diğer tüm faktörler projenin başarısını, gidişatını ve oluşan ürünü direkt olarak etkiler

Karmaşıklık, gelişim çevresel ve hedef çevresel değişiklerinin fonksiyonudur [3]. Projelerin karmaşıklığı arttıkça, kontrol güçleşir ve risk artar.

Klasik metodolojilerde, herhangi bir değişikliğin projeye getirdiği maliyetin projenin ilerleyen sürelerinde Şekil 2.1'de de gösterildiği gibi devamlı olarak artığını düşünürsek, değişiklikleri yönetebilmenin proje yönetimi uygulayıcıları için ne denli önemli bir kazanım olduğu ortaya çıkar.



Şekil 2. 1 Değişiklik maliyeti-zaman grafiği

3.2 Geleneksel Proje Yönetimi Süreçleri

Geleneksel proje yönetimi süreçleri (Şelale,V Model...) içerisinde analiz, tasarım, geliştirme, belgeme ve sürdürme aşamalarının olduğu, tahmin edilebilir süreçler olarak tasarlanmışlardır. Bu sert süreçler standart, yönetmelik ve yönergeler ile uyumlu karakterdedir. Projenin başarısı genellikle proje teslim standartlarına bağlıdır [4].

Süreç grupları birbiri ile bağımlı olarak çalışır. Yani bir süreç grubunun çıktı verisi bir diğer süreç grubunun girdi verisi olacağı için bir bağımlılık söz konusudur [5].

Geleneksel süreçler genellikle inşaat projelerinin yönetimi temelinde Şekil 2.2'de gösterildiği gibi adım adım doğrusal bir yapı temeline kurulmuştur [2].

Şekil 2. 2 Geleneksel süreçlerin lineer yapısı [2]

Geleneksel süreçlerin başarısızlığının tartışıldığı dönemlerde "Şimdiye kadar yazılıma inşaat gözüyle bakıldı; yazılım projeleri bina yapar gibi yapılmaya çalışıldı, o yüzden başarılı olunamadı. Bu şekilde başarılı olunamaz; yazılım inşaat gibi değildir ve yaklaşımı temelden değiştirmek lazım" fikri ortaya çıktı.

Bir inşaat projesine baktığınızda işin süreçleri ardışıktır:

Bir bina yapmaya karar verildiğinde önce binanın gereksinimleri, kullanım ihtiyacı, kat sayısı, inşaat alanı belirlenir. Ardından tasarım başlar. Tasarıma onay verilir ve bina inşa edilmeye başlanır.

Yazılım geliştirme projelerinde geleneksel metodlarla, öngörülebilir inşaat projelerinde olduğu gibi tüm değişkenler belirlenemez. Yazılım projeleri daha değişken yapıda, daha güvenilmeyen dinamik bir gereksinim yapısına sahip olabilir bu nedenle genellikle maliyet ve karmaşıklık değerleri eklenebilir [5].

Yazılım projeleri yönetimi ile inşaat projelerinin yönetimi farklıdır. İddia edilebilir sonuçlar üzerine güvenerek projeyi geliştirme yazılım proje gereksinimlerinin doğası gereği devamlı değiştiği için yazılım ekiplerinde stres yaratır. Yazılım projelerinde kararlı bir ortam olmadığından dolayı katı süreçler yazılım proje ekibini şaşırtır ve kargaşaya sürükler [6].

De Carlo'ya [5] göre geleneksel proje yönetimi süreçleri geçmiş zamanda organizasyonlarda çalıştı, ileride de çalışabilir. Günümüzde de bu süreçleri uygulayan ve işlerini bu yöntemler ile yöneten birçok organizasyon mevcuttur. Ancak yazılım

projelerinde devam eden bir başarılı olma baskısı varken ve geleneksel yöntemler bu isteklere cevap veremezken başarısız olunmasının temel sebepleri;

- Geleneksel yazılım süreçleri projenin başlangıcında müşterinin tüm gereksinimlerini tespit eder veya tespit ettiği kanısındadır, ona göre projeyi devam ettirir. Süreç yazılım geliştirme yaşam döngüsü içerisinde zamanla gelebilecek müşteri gereksinim değişikliklerine uygun yapıda değildir. Müşteri tarafından istenen değişiklikler hoş karşılanmaz, bu nedenle geliştirilen ürünün müşteri gereksinimlerini yüzde yüz tatmin etme olanağı yok gibidir.
- Müşterinin istediği değişikler zorunluluk nedeniyle yapılmak istenirse, süreç bunu ihtimal dahilinde saymadığı için değişiklik proje maliyetini yükseltir çünkü beraberinde yapısal değişiklikler getirebilir.
- Proje yöneticilerinin geliştiricilerden insan üstü beklentileri, projenin büyük bir bölümünün sorumluluk ve riskini omuzlarında taşıyan bu bireylerin fazla mesai yapmalarına, ruhen ve bedenen yorulmalarına neden olmaktadır. Motivasyonu düşük olan geliştiricilerin yaratıcı yönleri azalmakta ve kodun kalitesi motivasyona bağlı olarak düşmektedir.
- Geleneksel süreçler ile yürütülen projelerde takım çalışması ve bireyler arası iletişimin kuvvetlendirilmesi desteklenmez. Birbirinden uzak çalışan ve bilgi paylaşımı yapmayan geliştirici yazılımını yaptığı program parçasından sorumlu olduğu için, projeden ayrılan bireyler proje için risk haline dönüşür.
- Proje başlangıcında müşteri gereksinimleri en son detayına kadar tespit edilir. Ayrıca yazılım sistemi için teknik mimari ve uygulanacak tasarım belirlenir. Bunlar proje başlangıcında çok zaman kaybedilmesine neden olur. Ayrıca bir zaman sonra tasarımın, gerçeklenen gereksinimlere cevap vermediği ve yapılan her değişiklik sonucunda tasarımın çıkmaza girdiği görülebilir ve bundan dolayı başlangıçtaki çalışmalar gereksiz zaman kaybına neden olmuş olur.
- Oluşturulan ve uygulanan proje planı bir zaman sonra geçerliliğini yitirir. Plana mutlaka uyulması gerektiği düşüncesi, plan dışına çıkılmasını önlemek için fazla mesai

yapılmasını zorunlu kılar. Bu da motivasyonunu yitirmiş takım bireyleri olarak sonuçlanır.

Bu nedenle geleneksel yazılım proje yönetim süreçlerini kullanmak, güvenilir sonuçlar elde etme adına değişiklikleri tahmin ve yönetmede zorluk yaşatabilir. Bu problemden dolayı yazılım proje yöneticileri çevik süreçlerin yazılım proje yönetiminde daha başarılı olacağını, katı geleneksel yöntemlerin ise inşaat projeleri ve benzeri projelerde daha başarılı olacağı konusunda fikir birliğine vardı [5].

3.3 Yazılım Projeleri ile İlgili Anketler

Yazılım projelerinin başarı durumu ile ilgili yapılan anket çalışmalarına bakacak olursak projelerin başarısızlık oranı düşündürücü boyutlardadır;

Şekil 2.3' de görülen Standish Group tarafından yapılan Chaos anketine göre yazılım projelerinin başarı oranı aşağıdaki gibidir;

YIL	BAŞARILI (%)	EK MALİYET (%)	BAŞARISIZ (%)
1994	16	53	31
1996	27	33	40
1998	26	46	28
2000	28	49	23
2004	29	53	18
2006	35	46	19
2009	32	44	24

Şekil 2. 3 Chaos anketine göre proje başarı oranları

Standish Group tarafından yayınlanan 27 Nisan 2009 tarihli rapor, ilginç rakamlar içeriyor. 1985 yılından bu zamana bilgi teknolojileri sektöründeki eğilimleri tespit eden firmanın başkanı Jim Johnson 2009 Kaos Raporu'nda yaptığı açıklamada; planlanan zaman ve bütçede, tanımlanmış özellikleri ve işlevleri gerçekleştirebilmiş olan projelerin oranının %32' ye düştüğünü belirtiyor [7].

Projelerin %44' ü geç, bütçesini aşmış, tanımlanmış özellikleri ve işlevleri sağlamayacak şekilde tamamlanmış [7].

Geriye kalan %24'lük kısımdaki projeler ise tamamlanmadan iptal edilmiş veya tamamlandığı halde uygulamaya dahi alınmamış [7].

Standish Group CIO'su Jim Crear, başarısız projelerde geçen yıllara oranla büyük artış olduğunu belirtiyor. Başarılı proje oranı son on yılın en düşük seviyesinde bulunuyor. Yazılım projelerinin başarıya ulaşması, Standish Group raporunda da belirtildiği gibi her zaman kolaylıkla sağlanamıyor [7].

Başarı kriterleri temelinde yapılan başka bir Chaos anketinde Standish Group müşteri memnuniyetini ilk başarı faktörü olarak belirtiyor [8]. Çizelge 2.2'de başarılı olma kriterlerini görebiliriz.

Çizelge 2. 2 Chaos 1995 anketine göre projenin başarılı olma kriterleri

Başarı Kriteri	**Puan**
Kullanıcı katılımı	19
İdare desteği	16
Gereksinimlerin net açıklanması	15
Uygun planlama	11
Gerçekçi beklentiler	10
Daha küçük proje aşamaları	9
Yetkin çalışanlar	8
Sahiplik	6
Açık vizyon, hedefler	3
Çok çalışan odaklanmış ekip	3
Toplam:	100

Başka bir anket olan Shine Teknoloji 2003 anketi sonuçlarına göre çevik yöntemlere dayalı projeler geliştirildiğinde klasik yöntemlere göre verimlilik %93 artıp, %83 iş tatmini ve % 88 kalite artırımı sağlanmıştır [9].

Bir diğer çalışma da Scott W. Ambler tarafından 2010 yılında

- %20’ si geliştirici veya çevik takım üyesi, 33% 'si yönetici, takım lideri veya Scrum Master
- 77% ‘si IT sektöründe 10 yıldan fazla çalışan
- 20%’si 500’den fazla bilgi teknolojileri çalışanı olan organizasyonda çalışan
- %60’ı Kuzey Amerika, %22’si Avrupa, ve %10’u Asya Pasifik konumunda bulunan

203 katılımcı ile proje başarılarını araştırmak için gerçekleştirilen ankette şu sonuçlar elde edildi;

Şirketler belli kategorilerde başarıyı aşağıdaki gibi tanımlıyor;

Zaman-Plan

Şirketlerin %54’ü başarıyı ürünü zamanında teslim etme olarak tercih ederken, %44’ü sistem dağıtıma hazır olduğu zaman teslim etmeyi başarı olarak tanımlar.

Maliyet

Şirketlerin %35’i ürünü bütçe dahilinde teslim etmeyi başarı kriteri olarak tercih ederken, %60’ı projeler de iyi bir yatırım getirisi (ROI) sağlamayı tercih ediyor.

Fonksiyonalite

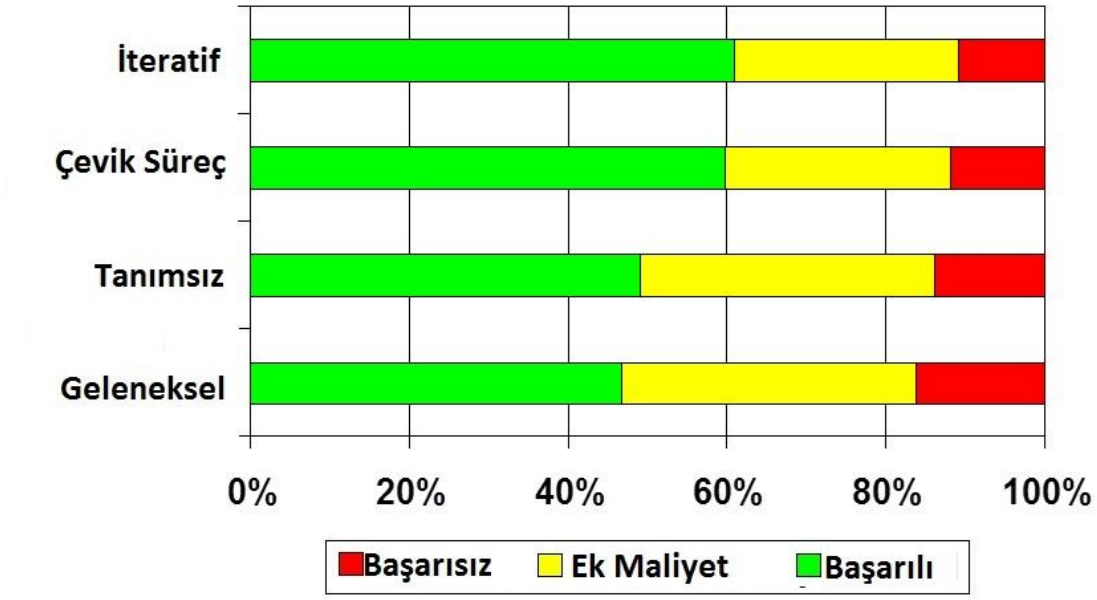
Şirketlerin %14’ü şartnameyi sağlamayı tercih ederken, %85’i paydaşların gerçek ihtiyaçlarını karşılamayı tercih ediyor.

Kalite

Şirketlerin %40’ı ürünü zamanında ve bütçe sınırları dahilinde sunmayı tercih ederken, %57’si yüksek kaliteli, kolay bakımlı sistemleri tercih ediyor.

Katılımcıların sadece %10’u başarı tanımlarını zamanında, maliyet sınırları dahilinde teknik şartnamelerine göre teslim etme olduğunu söyledi.

Süreçlerin başarı oranları ise bu ankette Şekil 2.4’deki gibi gözümüze çarpmaktadır;



Şekil 2. 4 Scott W. Ambler'in anketine göre proje başarı oranları

Bu çalışmalardan görüleceği üzere geleneksel süreçler artık başarılı olma yolunda iyi bir araç değil, değişiklik yapılması gerekiyor, kurumların ve kişilerin karakterleri, beklentileri ve yazılım geliştirme işi çevikliğe çoğu zaman daha yatkın ki bunu zaten çevik süreç kullanan organizasyonların başarılı proje oranlarından görüyoruz.

Klasik süreçlere alternatif olarak geliştirilen önemli süreçlere bakacak olursak;

3.4 Tümleştirilmiş Süreç (Unified Process)

Tümleştirilmiş Yazılım Geliştirme Süreci (Unified Process,UP) iteratif, artırımlı yapıda popüler bir yazılım geliştirme süreç altyapısıdır. En çok bilinen ve yaygın olarak kullanılan versiyonu Rational firması tarafından geliştirilen Rational Unified Process (RUP)'dir [10].

3.5 RUP (Rational Unified Process)

RUP iteratif, mimari merkezli, risk güdümlü ve kullanım senaryosu odaklı bir yazılım geliştirme yaklaşımıdır. İyi tanımlanmış ve yapısal bir yazılım geliştirme sürecidir ve özelleştirilebilir bir süreç altyapısı sağlayan bir süreç ürünüdür [11].

Rational firması tarafından geliştirilen RUP, organizasyon içindeki sorumlulukları detaylı olarak belirleyerek disiplinli bir yazılım geliştirme süreci sunar. RUP içinde yazılım yaşam çevrimi 4 aşamadan oluşmaktadır [11].

Başlangıç aşamasında; yaklaşık tahminler yapılarak temel gereksinimler ortaya konulmaya çalışılır. 2. aşamada daha gerçekçi tahminler yapılarak mimari belirlenir ve yüksek riskler için çözümler sunulur. 3. aşamada, ürün hazır hale getirilerek hatalar belirlenir. 4. aşama ise müşteriye ürünün teslim edildiği ve müşteriden gelen geri beslemeye göre iyileştirmelerin yapıldığı aşamadır [11].

Her aşama bir durak noktası ile sonlanmakta ve her aşama sonunda bir değerlendirme yapılarak amaçlara ne oranda ulaşıldığı incelenmektedir [11].

3.5.1 Başlangıç

Neyin geliştirileceğinin anlaşılması kısmıdır. İstenilenlerin bir ilk değerlendirilmesinin yapılması ve ne elde edilebileceği konusunda anlaşmaya varılması bu aşamada gerçekleştirilir. Amaç iş gereksinimlerini toplamak ve anlamak, proje planını detaylandırmak ve işin büyük resmi üzerinde mutabakat sağlamaktır. Bu fazda projenin öncelikli hedefleri, varsayımları, kısıtları, gerçeklenecek özellikleri ve kabul kriterleri tanımlanır [11].

3.5.2 Detaylandırma

İşin nasıl gerçekleştirileceğinin tanımlanması kısmıdır. Bu aşamada sistem mimarisi tasarlanır. Hedef gereksinim ve özellikleri teknik çözüm haline getirebilmektir. Yüksek seviye tasarım projenin genel mimarisi ve altyapısı ile ilgilenirken, alt seviye tasarım projenin daha detay özellikleri ile ilgilenir [11].

3.5.3 İnşa

Ürünün geliştirildiği kısımdır. Burada vurgu maliyet, zaman ve kalite üzerinedir. Komple sistemi geliştirmek için adım adım için çalıştırılabilir mimarisi geliştirilmeye çalışılır. Tamamen işlevsel bir yazılım sunumun çıkılması hedeflenir [11].

3.5.4 Ürün Geçişi

Çözümün onaylanması kısmıdır. Bu aşama, ürünün sürüm hazırlıklarının, ürüne ait ayarların, yükleme ve kullanılabilirlik konularının testlerinin yapıldığı aşamadır. Burada dikkat edilen husus ürünün son kullanıcılar için uygun olup olmadığıdır [11].

Dikkat edilmesi gereken önemli bir nokta; RUP içindeki her aşamada yazılım mühendisliğinin tüm aktivitelerinin farklı oranlarda uygulanıyor olmasıdır. Klasik yazılım geliştirme süreçlerinde, her aktivite bir önceki aktivitenin bitmesinin ardından gerçekleştirilmektedir.

RUP'da sistem aşağıdaki disiplinlerde (iş akışları) geliştirilir [11];

Disiplin-1 İş Modelleme: Organizasyon yapısını anlamak sistemin geliştirilmesi için önemlidir.

Disiplin-2 Gereksinimler: Projenin kapsamı belirlenir. Projenin fonksiyonel (iş kuraları, kullanıcı arayüzü) ve fonksiyonel olmayan özelliklerinin belirtildiği özellik dokümanı hazırlanır.

Disiplin-3 Analiz ve Tasarım: Gereksinimler analiz edilir ve sistemin mimari tasarımı yapılır. Bu aşamada ağ, veritabanı ve modül tasarımı yapılır.

Disiplin-4 Gerçekleştirme: Kodlar geliştirilir ve birim testleri yapılır.

Disiplin-5 Test: Geliştirilen sistemin kalitesi ölçülür. Bu kapsamda hata tespiti, sistemin tasarımda belirtilen şekilde çalışıp çalışmadığı, özellik dokümanında belirtilen gereksinimlerin tümünün karşılanıp karşılanmadığı gibi durumlar kontrol edilir.

Disiplin-6 Dağıtım: Bu aşamada yazılımın teslimatının planlanması ve gerçekleştirilmesi, dağıtım dokümantasyonunun hazırlanması ve sistemin son kullanıcılar tarafından kullanılmaya hazır edilmesi gibi aktiviteler gerçekleştirilir.

Disiplin-7 Konfigürasyon ve Değişiklik Yönetimi: Değişen gereksinimlerin uygulanıp yönetimi, konfigürasyon öğelerinin yönetimi ve sürüm yönetimi gibi konular ele alınır.

Disiplin-8 Proje Yönetimi: Görev atamaları, risk yönetimi, süreç takibi gibi konuların ele alınarak projenin zamanında ve belirtilen bütçede tamamlanmasını sağlama faaliyetleri yürütülür.

Disiplin-9 Ortam: Gerekli olduğu durumlarda kullanılması gereken araçların uygun olup olmadığının kontrolü

Süreç 4 statik modeli kullanır bunlar [11];

• Çalışanlar, kimi

• Aktiviteler, nasılı

• Ürünler, neyi

• İş akışları, ne zamanı cevaplar.

RUP aşağıdaki pratikleri kullanır [11];

•İteratif yazılım geliştirme

•Gereksinimleri yönetme

•Bileşen tabanlı mimari kullanımı

•Görsel modelli yazılım

•Sürekli yazılım kalitesi kontrolü

•Yazılım değişimlerini kontrol etme

Sürecin iteratifliğinin katkıları [11]:

•Değişen isteklere uyum

•Erken geri besleme

•Büyük sistemlerde çözümleme kolaylığı

•Risklerin erken sezilmesi ve giderilmesi

•Yeniden kullanılabilirlik

•Erken ürün elde etme

•Hataları birkaç iterasyonda saptama ve düzeltme

•Her iterasyonda deneyim kazanma

3.6 Çevik Süreçler

Çevik süreçler yazılım sektöründe kullanılan mevcut geleneksel yöntemlere alternatif olarak geliştirilmiş, modern ve bürokrasiye mesafeli yazılım yöntemlerini ihtiva ederler.

3.6.1 Tarihçe

Çevik yazılım süreçleri, 1950'lerdeki üretim alanında verimliliğin artırılması için geliştirilen yalın yaklaşımların, yazılım sektöründe bir uzantısı olarak ortaya çıkmıştır [1].

Yazılım geliştirme için daha başarılı yollar bulmak üzere çalışan insanlar bazı değerleri ön plana çıkarak daha başarılı ve kaliteli yazılımın ortaya çıkacağı fikrini ortaya atmışlardır.

2001 yılında Utah'da, bu düşüncedeki 17 kişi bir araya gelip çevik yazılım geliştirme manifestosunu hazırladı.

Bu bildirgeye göre [12];

1.Kişiler ve iletişim süreç ve araçlardan önce gelir.

2.Çalışır durumda olan program detaylı dokümantasyondan daha önceliklidir.

3.Müşteri ile beraber çalışmak sözleşmelerden ve anlaşmalardan daha önceliklidir.

4.Değişikliklere ayak uydurmak bir planı takip etmekten daha önemlidir.

Bu öğeler dışında belirlenen çevik süreç prensipleri de [12];

- En önemli öncelik erken ve sürekli olarak kullanılabilir programlar oluşturarak, müşteriyi tatmin etmektir.
- Yazılımın ilerleyen dönemlerinde gelse bile talep edilen değişiklikler hoş karşılanmalıdır.
- Kısa sürelerde (birkaç haftadan, birkaç aya kadar sürebilen zaman dilimlerinde) çalışır programlar ortaya konulmalıdır. Seçim, zaman diliminin kısa tutulması yönünde olmalıdır.
- Müşteri ve geliştiriciler proje süresince beraber çalışırlar.
- Projelerin motivasyonu yüksek bireyler tarafından yapılması sağlanmalı, onlara ihtiyaç duydukları ortam, destek sağlamalı ve işi bitirebileceklerine inanılmalıdır.
- Bilgi alışverişinde en verimli ve etkili yöntem takım içinde yüz yüze konuşmaktır.
- Çalışır durumda olan program ilerlemenin ana göstergesidir.

- Çevik süreçler etkili yazılım yöntemlerini destekler. Müşteri, programcılar ve kullanıcılar sabit bir tempoda beraber çalışabilmelidirler.
- Devamlı teknik mükemmelliğe özen gösterilmesi ve iyi tasarım, çevikliği kuvvetlendirir.
- Sadelik (basitlik) esastır.
- En iyi mimariler, gereksinimler ve tasarımlar kendi kendine organize olabilen takımlardan çıkar.
- Belirli zaman dilimlerinde takım daha nasıl etkili olabileceği konusunda kendini sorgular ve edindiği bilgiler doğrultusunda çalışma tarzını adapte eder.

Çevik metodolojilerin uygulandığı projelerde takımlar kısa geliştirme döngüleri için de (zaman kutusu olarak da bilinir) müşteri veya kullanıcıya çalışan ve değer katan bir yazılım sunmak için anlaşırlar [13].

Yazılım dünyasında çeşitli çevik yaklaşımlara 1970'lerden itibaren rastlanabilmekle birlikte, çevik yazılım metodolojilerinin kullanımı 1990' larda hız kazanmış ve geçtiğimiz son 7-8 yıl içerisinde de tüm dünyada başarılarını kanıtlayarak organizasyonlar tarafından kullanımı arttırmıştır. Şu anda, dünyadaki birçok yazılım şirketinde ve birçok yazılım projesinde yazılımlar, çevik yaklaşımlarla geliştirilmektedir [1].

3.6.2 Özellikler

Çevik yazılım geliştirme metodu, tekrarlanan yazılım geliştirme metodu taban alınarak geliştirilmiş, sık aralıklarla parça parça yazılım teslimatını ve değişikliği teşvik eden bir yazılım geliştirme metodolojisidir [1].

Çevik geliştirme [1];

- değişimi,
- takım içerisindeki iletişimin arttırılmasını,
- parça parça yazılım teslimatını,
- test odaklı yazılım geliştirilmesini,

- uyumlu planlamayı teşvik eder.

Çevik yazılım (Agile Development) bir yandan bir değer sistemini, diğer yandan da somut yazılım metotlarını içerir. Çevik yazılıma, yazılım sektöründe yeni bir felsefi akım ya da yeni bir yazılım meta modeli olarak bakılabilir. Bu yeni akımın somut örnekleri arasında Uç Programlama (XP, Extreme Programming), Scrum ve Lean Development bulunmaktadır.

Yazılım geliştirme sürecinin ilerleyen aşamalarında ortaya çıkacak değişim isteklerinin getireceği maliyet nedeniyle, geleneksel yaklaşımlar, başlangıçta yeterince kapsamlı çalışıp ihtiyaçları eksiksiz tespit etmeyi ve bu sayede projenin ileri safhalarında ortaya çıkabilecek olası değişimleri önlemeyi esas alır. Ancak hızlı gelişen teknoloji ve artan rekabet ortamında müşteri gereksinimleri artık daha hızlı değişmekte ve değişim isteklerinin daha çabuk yerine getirilmesi gerekmektedir. Proje başlangıcında eksiksiz tespit edilmeleri imkansızlaşan müşteri ihtiyaçlarındaki değişimlere engel olunamadığına göre, yazılım projelerinde kullanılacak yazılım geliştirme yöntemlerinin de bu değişimleri daha kısa sürede ve daha az maliyetle karşılayabilecek özelliklere sahip olması beklenmektedir [14], [15].

Değişime adapte olmak, geç bi sürede de olsa müşteriden yeni gereksinimler kabul edebilmek anlamına gelir [16].

Ağır yazılım geliştirme yöntemlerinde bürokrasinin yoğun olması, işlerin çok yavaş ilerlemesi, yazılımcı yeteneklerinin üst seviyede kullanılamaması ve yazılım mühendislerinin gerçek işlerini (tasarım, analiz, kodlama vb.) yapamaması söz konusu olmaktadır.

Çevik süreç yönetimi, proje yönetimi uygulayıcılarına geleneksel yönetim süreçlerinde katı bağımlılık nedeniyle mümkün olmayan beklenmedik sorunlar karşısında çözüm üretimine yardımcı olması için prensipler, pratikler ve değerlerden oluşur [17].

Çevik proje yönetiminin temel özelliği geleneksel metodların aksine, değişimi kontrol etmekten daha çok onu kabullenme, azaltma ve eleme yeteneği kazandırmasıdır [18].

Çevik metodları farklı kılan esas unsurlar ise aşağıdaki gibi sıralanabilir:

Takım Halinde Planlama

Çevik yaklaşımların ilk özelliği, süreçler ve araçlardan çok geliştirme ekibindeki kişilere değer vermesidir [19]. Çevik metodlar, tek başına (proje yöneticisi) değil birlikte yapılan işbirliğine dayalı planlamayı vurgular. Örneğin ilk başlarda yapılacak sürüm planlaması veya her yineleme planlaması toplantıları ideal olarak bütün geliştiricilerin ve müşterilerlerin katılımı ile yapılır. Eğer proje büyük ve alt takımlardan oluşuyorsa, en azından her takımdan bir temsilcinin bu toplantılara katılması sağlanır.

Herkesçe Ulaşılabilir Görülebilir Proje Planları

Proje planları (basit veya detay planlar) tüm takımın kolayca görebileceği şekilde sergilenir.

Süre Kestirimini İşi Yapacak Kişi Yapar ve O İşi Gerçekleştirmeyi Taahhüt Eder

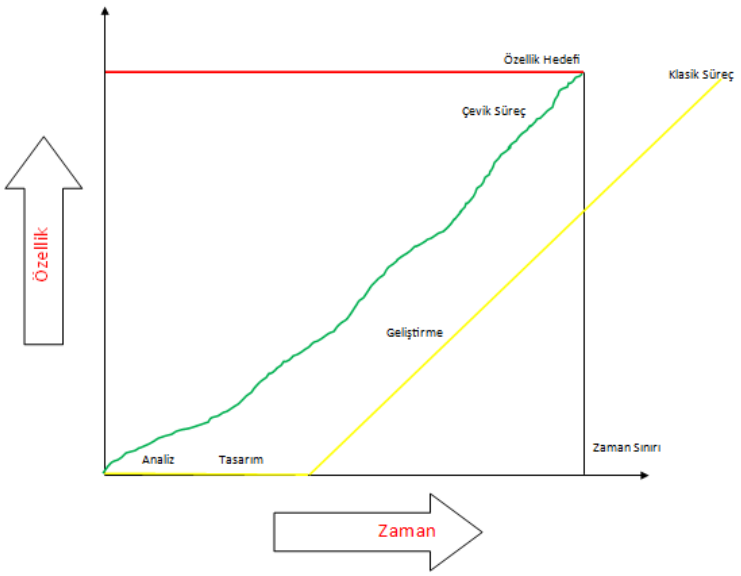
Çevik metodlar, yapılacak işlere ait tahmini süre kestiriminin bizzat o işi yapacak kişilerce yapılmasını izin verir. Bireyler kendi kendine organize olur.

Gönüllülük Esası

Çevik metodlardan olan XP ve Scrum yapılacak görevlerin yönetici tarafından atanması yerine, takım üyelerinin istedikleri görevlere kendilerinin talip olmalarını vurgular.

Çevik proje yönetimi proje yöneticilerin daha kısa sürede başarılı ürüne ulaşabilmesi için, dinamik ve uyarlanabilir şekilde üretilebilirlik ve erken müşteri geri dönüşü alabilme yeteneği sağlar [18].

Erken müşteri dönüşü için klasik süreç ile çevik süreçlerin karşılaştırılması Şekil 2.5'deki gibi görünebilir.



Şekil 2. 5 Geleneksel modeller ile çevik modellerin karşılaştırılması

Yazılım geliştirmede uygulanan yöntemin başarısı, takip edilen sürece ve kullanılan araçlara bağlı olsa da sonuçta süreci uygulayan ve araçları kullananlar insanlardır. Yetenekli ve usta ekiplerle uygulanan süreçlerin başarısız olma olasılığı düşüktür. Çevik yaklaşımlar, çalışanlara gerekli ortamı ve desteği sağlayarak, başarılı olacaklarına güvenmeyi ve kararlarına değer vermeyi gerektirir.

Bu akımın öncü ürünlerinden XP test güdümlü programlama, basit tasarım gibi programlama pratiklerine yoğunlaşırken, Scrum planlama üzerine yoğunlaşır. Her ikiside Çevik Manifesto'yu temel alır. Bu süreçler takımlar için yüksek motivasyon ve işleri kolaylaştırma adına her yinelemede gelişim sağlamayı hedefler [13].

3.6.3 Çevik Süreçler ile RUP'in Karşılaştırılması

İki süreç birçok ortak özelliğe sahip olsada aşağıdaki gibi farklılıklara sahiptir [20];

- RUP, daha çok büyük ölçekli projelerde tercih edilmektedir.
- RUP, değişiklik maliyetinin üstel olarak arttığını varsaymaktadır. Bu nedenle risk azaltma konusu ile çok fazla ilgilenmektedir. Çevik yazılım ise, geliştirme aşamasında değişiklik maliyetinin düşük olduğunu varsayar. Bu nedenle mimarinin zamanla olgunlaşmasına izin verir.

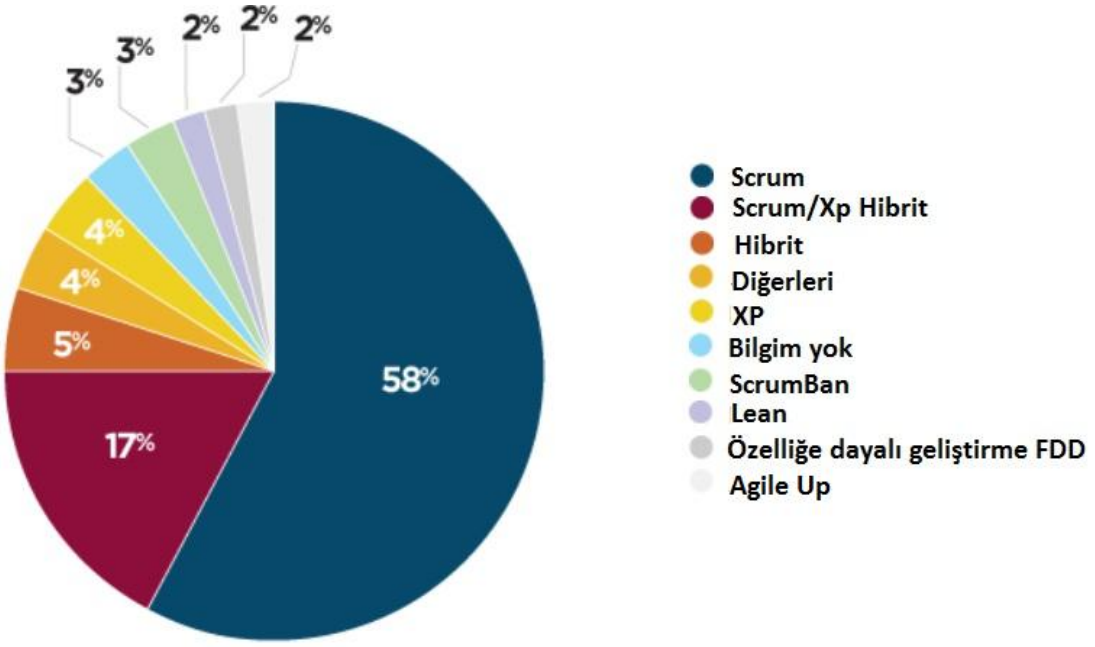
- RUP'un uygulandığı projelerde, ekipler daha fazla eleman içerir, Çevik süreç küçük gruplarla çalışmayı önerir.
- Çevik süreçlerde daha fazla özyineleme vardır ve özyineleme süreleri daha kısadır.
- Çevik süreçler, RUP içindeki tüm disiplinleri içermez.
- Çevik süreçlerde çalışanların sorumlulukları daha fazladır.
- Çevik süreçler ile RUP rollerinde farklılıklar vardır.
- Çevik süreçlerde geliştirilen kod üzerinde ortaklaşa sahiplik söz konusudur. Herhangi bir anda ihtiyaç duyan, kodun bir parçasını değiştirebilir. RUP'da ise herkes rolünün gereğini yapması gerekir.
- RUP, çevik süreçlere göre mimariye daha fazla önem verir.
- Çevik süreçler, bir doküman kullanılmıyorsa defalarca güncellenmesini istemez. Ayrıca, çok fazla sayıda belge oluşturmanın faydasına inanmaz. RUP ise tasarımın sürekli güncel olmasını ister. Çevik süreçlerde yazılan belge yerine kodun etkin olarak çalışmasını arzular.
- RUP'daki bazı sorumluluklar çevik süreçler içinde bulunmayabilir çünkü ilgili aktivite süreç içinde yer almıyor olabilir.

3.6.4 Çevik Süreçler ile İlgili Yapılan Anket Çalışmaları

Çevik süreçler ile ilgili yapılan anket çalışmalarına bakıldığı zaman;

VersionOne sponsorluğunda gerçekleştirilen 2010 yılında 91 ülkede yaklaşık 2700 katılımcı ile gerçekleştirilen "State of Agile Development" isimli ankette ortaya çıkan sonuçlara göre organizasyonların çevik süreçlere bakış açılarını şöyle özetleyebiliriz;

Şekil 2.6'da organizasyonlara hangi çevik süreci kullandıkları sorusuna verilen cevapları görüyoruz. En fazla kullanılan sürecin Scrum olduğu grafikten görülmektedir.



Şekil 2. 6 Organizasyonların kullandığı çevik süreçler [21]

Çizelge 2.3'de çevik tekniklerin kullanım sıklığını göstermektedir. İteratif olarak plan yapma ve gerçekleştirme ile günlük toplantı yapma genellikle organizasyonların başarılı bulduğu ve yüksek oranda tercih etttiği tekniklerin başında geliyor.

Çizelge 2. 3 Organizasyonların kullandıkları çevik teknikler

İteratif planlama	83%
Günlük toplantı	82%
Birim test	77%
Dağıtım planlama	72%
Geriye bakış	68%
Azalan zaman	67%
Devamlı integrasyon	65%
Otomatik kurulum	60%
Hız	57%
Refaktör	57%
Kod standartları	56%

Test güdümlü programlama	46%
Açık çalışma alanları	43%
Dijital görev panosu	37%
Toplu kod sahipliği	36%
Çift programlama	33%
Otomatik kabul testi	31%
Müşteriyle beraber çalışma	29%
Sürekli dağıtım	25%
Gerçek zamanlı üretim	18%
Zaman döngüsü	12%
Özellik temelli geliştirme	9%

Çevik süreçte karşılaşılan başarısızlık sebebi ile ilgili cevaplar Çizelge 2.4'de görülmektedir. En büyük sebebin tecrübe eksikliği olduğu görülmektedir.

Çizelge 2. 4 Çevik projelerin başarısız olma sebepleri

Başarısız bir proje deneyimi olmadı	22%
Çevik süreçlerde tecrübesiz olmak	14%
Çevik değerler ile şirket kültürü ve değerlerinin çatışması	11%
Diğer	11%
Geleneksel metodlar ile çalışmak için dış baskı	10%
Kültürel geçiş eksikliği	8%
Yönetim destek eksikliği	7%
Çevik pratikleri takip etmede takımın isteksizliği	6%
Süreçte yeni olmak	6%

Yetersiz eğitim	5%

Çizelge 2.5'de bir organizasyonda çevik sürecin kabulü için ortaya çıkan engeller listelenmiştir. En önemli engeller aşağıda görüldüğü gibi orgazisyon kültürünü değiştirme zorluğu, değişime karşı direnç ve yetkinlik eksikliğidir.

Çizelge 2. 5 Çevik süreç kabulu için ortaya çıkan engeller

Organizasyon kültürünü değiştirme yeteneği	51%
Değişime karşı olan direnç	40%
Gerekli becerilere sahip personel imkanı	40%
Yönetim desteği	34%
Proje karmaşıklığı veya boyutu	31%
Özel işbirliği	29%
Çevik ölçüm yeteneğine güven	25%
Geçiş için algılanan zaman	16%
Maliyet kısıtları	13%
Bilinmiyor	12%
Diğer	6%

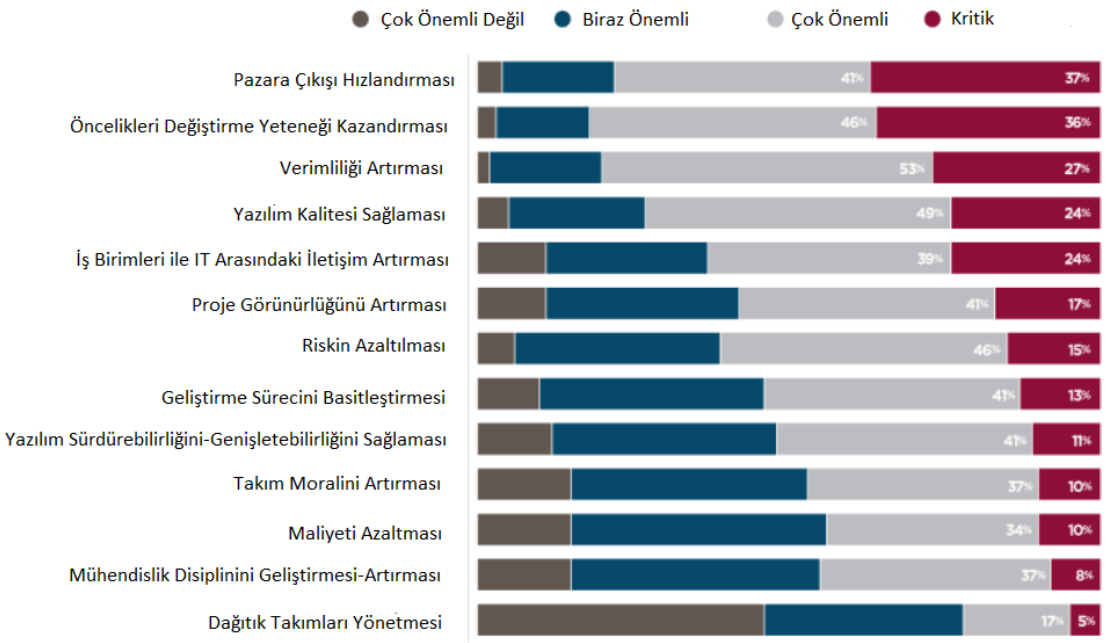
Çizelge 2.6'da organizasyonların çevik sürece geçerken yaşadığı kaygılar sorusuna verilen cevaplar listelenmiştir. Bu kaygıları aşmaya yardımcı olmak çevik süreçlere geçişleri kolaylaştıracaktır.

Çizelge 2. 6 Organizasyonların çevik sürece geçerken yaşadıkları kaygılar

Yönetim kontrol kaybı	36%
Ön planlama eksikliği	33%
Yönetim değişimi karşıtlığı	32%
Döküman eksikliği	28%
Öngörülebilirlik eksikliği	27%

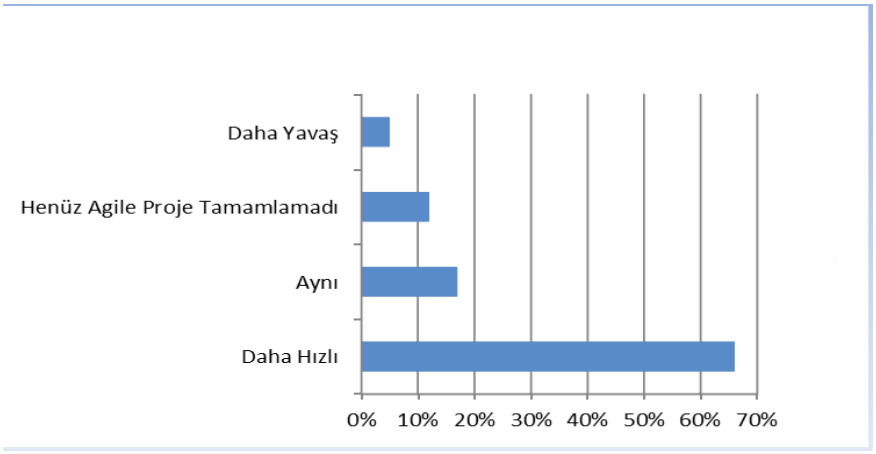
Mühendislik disiplini eksikliği	25%
Geliştirme takımının değişikliğe karşıtlığı	21%
Mühendislik yetenek kalitesi	16%
Diğer	15%
Ölçme yetersizliği	12%
Mevzuata uygunluk	12%
Azaltılmış yazılım kalitesi	11%
Fikir sahibi değil	11%

Organizasyonların çevik süreçlerde neler kazandığını öğrenmek adına sorulan sorulara alınan cevaplar Şekil 2.7'de görüldüğü gibidir.



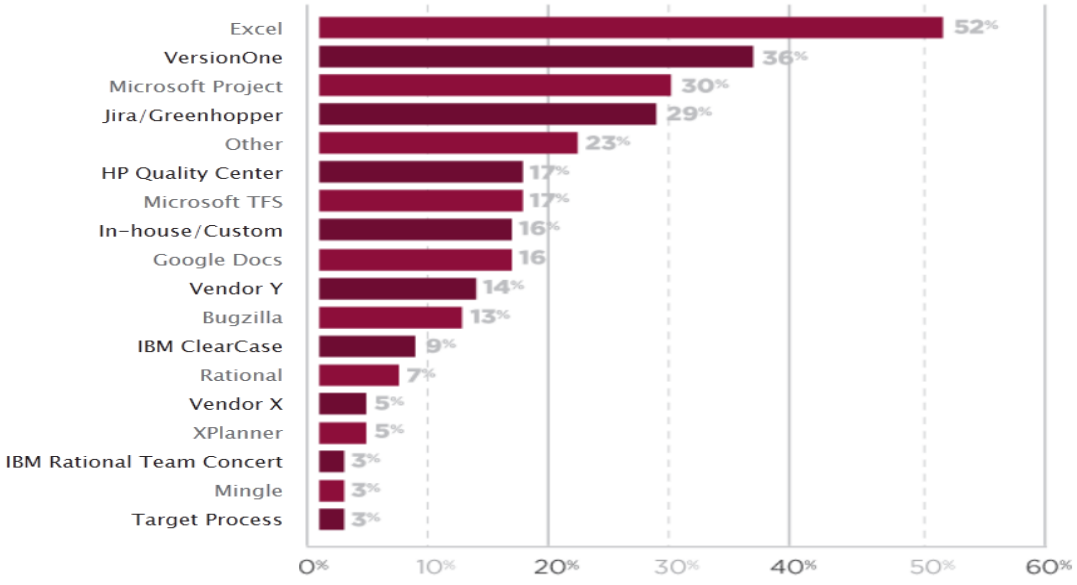
Şekil 2. 7 Çevik süreçlerin organizasyonlarına kazandırdıklarının önem listesi [21]

Organizasyonların çevik sürece tercih ettikten sonra eski durumlarına göre başarı sağlayıp sağlayamadıklarına dair cevaplar Şekil 2.8'de görülebilir. Bu sonuç %60'ın üzerinde daha başarılı olunduğunu gösteriyor.



Şekil 2. 8 Eski yöntemleri ile çevik süreçlerin farkı [21]

Yapılan araştırmada organizasyonların proje yönetimi için kullanılan araçlara verdiği cevaplar görülebilir (Şekil 2.9) . Burada Excel'in yüksek kullanım oranını görmek bize doğru yolda olduğumuzu anlama adına motive kaynağı oldu.



Şekil 2. 9 Proje yönetimi için kullanılan yazılım araçları [21]

Bu anket çalışması gösteriyor ki kişiler çevik süreçlerden en çok Scrum'ı ve onun pratiklerini kullanıyorlar ve bireylerin %80'inden fazlası projeyi uygun olmayan araçlar ile yönetmeye çalışıyor.

3.7 Çevik Süreçlere Uyum Kriterleri

Bir yazılım geliştirme yönteminin her tür projede iyi sonuç vermesi beklenmemelidir [22].

Yazılım projelerinde kullanılacak yazılım geliştirme yöntemini belirlerken projenin özelliklerini dikkate almak gerekir. Projeye uygun olmayan yöntemin kendisi bazen başarısızlık sebebi olabilmektedir. Şu halde, belli metodolojilerin veya süreçlerin ne zaman ve hangi şartlar altında en iyi çözüm olacağını ve bulunulan duruma göre nasıl biçimlendirileceğini bilmek önemlidir.

3.7.1 Ekibin Büyüklüğü

Çevik yöntemlerde çalışanlar arasında iyi iletişim ve etkileşim şarttır. Bu nedenle çevik yöntemlerin uygulanması kararını etkileyen belki de en önemli faktör çalışanların sayısıdır. Çevik yöntemleri deneyen proje ekipleri çoğunlukla onbeş kişiyi geçmemektedir [23], [24].

Kalabalık geliştirme ekiplerinde yüz yüze iletişim ve koordinasyon zorluğu, çevik yöntemlerin başarısını olumsuz yönde etkileyecektir. Bu durumda ekibi küçük takımlara bölmek gibi ölçekleme yöntemleri uygulanmalıdır [23].

Çevik yaklaşımlarda yüzyüze görüşme ve müşteriyle birlikte çalışma ihtiyacı, proje ekibinin aynı ortamda bulunmasını gerektirir. Birbirinden farklı uzak mekanlarda bulunması gereken çalışanlar aynı takımda yer almamalıdır [25].

3.7.2 Ekipteki Ustalık

Çevikliğin özünde, tarifi gereksiz bilgileri planlara yazmak yerine ekibin kendiliğinden biliyor olmasını beklemek yattığından, çevik yöntemler kullanılacaksa, geliştirme takımında tarif gerekmeyen yetenekli ve tecrübeli kişiler bulunması gerekir [22].

Proje ekibinin hepsi olmasa bile bir kısmının geçmişte benzer projeler geliştirmiş, kullanılacak teknolojiye hakim ve insan ilişkillerinde iyi özelliklere sahip olması, çevik yöntemlerin uygulanabilirliği için önemlidir [25].

3.7.3 Müşteri Profili

Çevik yaklaşımların genel prensibi, müşterinin de geliştirme ekibi ile birlikte aynı takım içinde çalışmasını gerektirir. Başlangıçta detaylı gereksinim analizi yapılmadığından,

tasarım ve kodlama yapılabilmesi için analizdeki eksikleri doldurmak üzere müşteriyle birlikte çalışmak süreç için önemlidir [25].

Müşteri profilinin buna uygun olmaması halinde çevik yöntemler başarısız olacaktır. Örneğin ihtiyaçların, müşterinin üst yönetimi tarafından bilinmesi ve bu kişilerin de zamanlarının kısıtlı olması nedeniyle birlikte çalışma mümkün olmayabilir. Müşteri uzak bir ortamda bulunabilir veya birlikte çalışmaya istekli olmayabilir [25].

3.7.4 Kritik Uygulamalar

Çevik yöntemlerin; insan hayatını etkileyen sistemler, güvenlik uygulamaları, nükleer sistemler, uzay çalışmaları ve silah sistemleri gibi kritik ve başarısızlık maliyeti yüksek projelerde uygulama örneğine rastlanmamıştır. Yüksek güvenilirlik beklenen bu tip projeler çok iyi test edilmeden teslim edilemez. Çevik yaklaşımlarda uygulanan iteratif ve kademeli geliştirme yönteminin yeterli sıklıkta test ortamı yarattığı düşünülebilir [23].

3.7.5 Bakım Safhası

Bazı projelerde bakım safhası, geliştirme safhasından daha önemlidir. Proje teslim edildikten sonraki süreçte geliştirme ekibinde olası değişiklikler ve dokümantasyon eksiklikleri nedeniyle bakım işlemleri maliyeti arttırarak projenin yeniden yapılmasına bile yol açabilmektedir. Çevik yöntemlerde, “gerektiği kadar dokümantasyon” ilkesi olması sebebiyle, bakım safhasında lazım olacak dokümanlardan feragat edilme riski bulunur. Bu nedenle projenin tesliminden sonraki aşamalar da düşünülerek bakım safhasında gerekli olabilecek dokümantasyon hazırlanmalıdır [25].

Çevik yazılım geliştirme yöntemlerinin bakım süreçlerini nasıl etkileyeceği henüz fazla gündeme gelmemekle [24] birlikte, farklı bir ekip tarafından daha önce çevik yöntemlerle geliştirilmiş bir uygulamanın bakım projesi, uygulama hakkında yeterli bilgi ve belge olmazsa bilinen yazılım mühendisliği sorunlarıyla karşı karşıya kalacaktır [25].

3.7.6 Çevikliğe Yatkınlık

Çevik yaklaşımları kullanmayı düşünenlerin, öncelikle kurumları içindeki anlayış ve alışkanlıkları çevik yaklaşımlara uydurmaları gerektiği savunulmaktadır [26].

Çevik yöntemler, yazılım geliştirme teknikleri ve zamanlama gibi teknik konularda karar verme yetkisini üst yönetimden alıp geliştirme ekibinin sorumluluğuna verdiklerinden [27], kurum içinde bu değişimin benimsenmiş olması gereklidir.

Projenin kendisi çevik yöntemlerin uygulanmasına çok elverişli olabilir. Ancak, geliştirme ekibine yeterince serbestlik tanınmayan, ekibin çalışma şekillerinin katı kurallar ve yöntemlerle kısıtlandığı ortamlarda çevik yaklaşımların başarı şansı azalacaktır [25]. Müşteriye, proje başlangıcında gereksinimleri biçimsel bir şekilde imza altına almayan bir sözleşmeyi kabul ettirmenin zor olacağı dikkate alınarak, müşterilerin de çevik yöntemler konusunda eğitilmesi gerekebilir [26].

BÖLÜM 3

SCRUM

Scrum varolan bir sistem veya yeni bir ürün prototipi için yönetme, geliştirme ve bakım metodolojisidir [3]. Yüksek kaliteli yazılımın sık ve düzenli teslimi boyunca organizasyonlara iş değerinin erken ve doğru teslimine izin veren bir süreç yapısıdır.

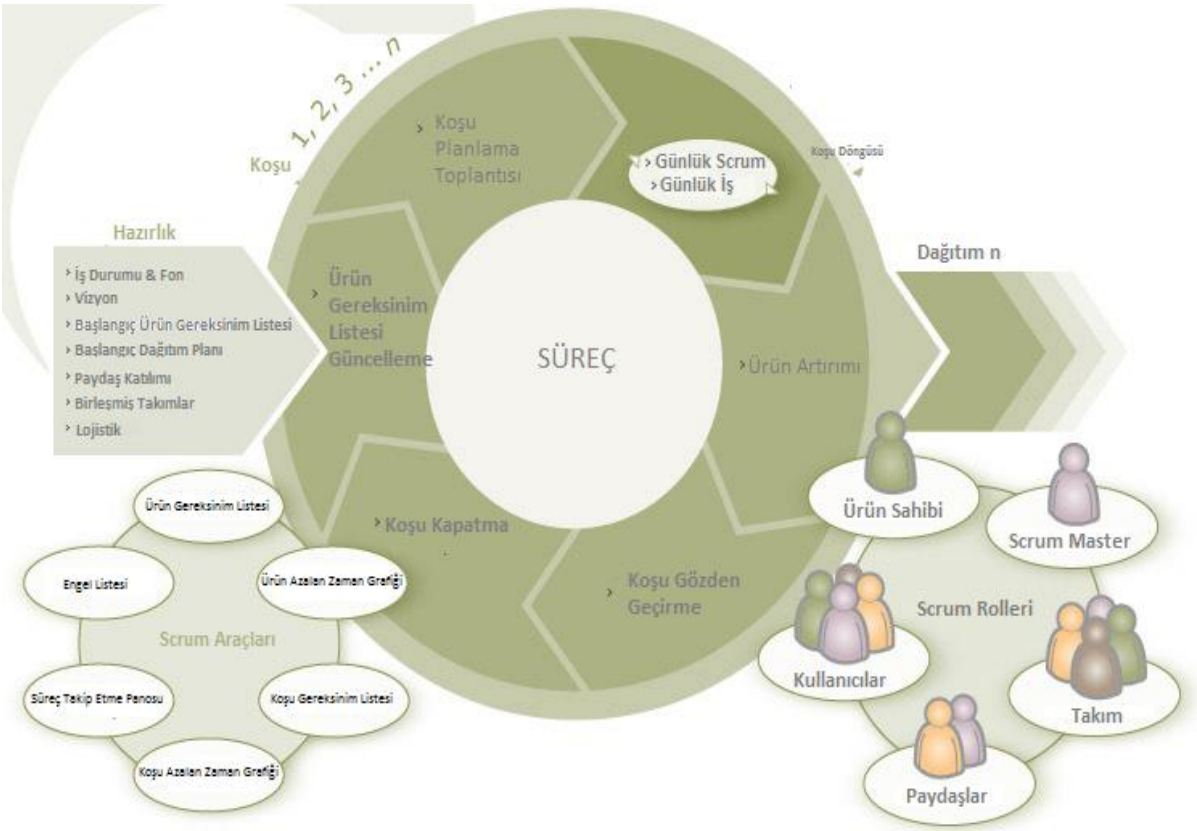
Scrum on yıldır süre gelen endüstride de kabul edilmiş, kullanılan ve sağlanan en başarılı pratikler temeline dayandırılmış, tecrübeye dayanan süreç teorisinden meydana gelmiştir [28].

Scrum, iteratif ve artırımlı nesne tabanlı geliştirme döngüsü içerisinde kullanılan gelişmiş bir süreçtir [3].

Scrum, XP, Özelliğe Dayalı Geliştirme (Feature Driven Development, FDD), Crystal gibi çevik yazılım geliştirme yaklaşımlarını benimser. Karmaşık yazılım ve ürünü basamak basamak geliştirerek kontrol ve yönetmeyi amaçlayan bir süreçtir.

Gereksinimleri sağlayacak, başarılı ve kaliteli bir yazılım üretimi için bir araya gelmiş takım ve müşteri birlikteliği Scrum'ın getirdiği en büyük avantajlardandır.

Sürecin genel görünümü Şekil 3.1'den görebiliriz;



Şekil 3. 1 Scrum süreci genel görünümü [29]

4.1 Tarihçe

1986 yılında, Hirotaka Takeuchi ve Ikujiro Nonaka ürün geliştirmede bütün süreç evrelerinin birbiriyle örtüştüğü ve takımın farklı evrelerde de birlikte çalışabildiği bir yeni bir yaklaşım tanımladı. Bu yeni yaklaşımın temeli otomotiv, bilgisayar, fotokopi makinesi, yazıcı üreten firmaların vaka çalışmalarına dayanır ve temel olarak üretime hız ile esneklik getirmek amacı vardı. Onlar bu yaklaşıma holistik ve rugby ismini vermişti, tüm süreç çapraz görevli ve tek bir amaç için çalışan takım tarafından yürütülürdü [30].

1990'ların başlarında Ken Schwaber kendi şirket bünyesinde Easel şirketi için bu yaklaşımı uyguladı ve ona Scrum adını verdi [31].

1995 yılında ise Sutherland and Schwaber ortaklaşa Teksas'daki bir çalışmada Scrum metodolojisini anlatan, ilk halka açık sunum özelliğini taşıyan yazıyı sundular [32].

Schwaber ve Sutherland şimdilerde Scrum olarak bilinen süreci zaman için de yazıları, kendi deneyimlerini sektördeki uygulamalar ile pekiştirerek insanlara sundu.

Scrum, yüksek seviyede belirsizlik arz eden projelerde son yıllarda yoğun biçimde uygulanan ve başarılı sonuçlar üreten bir tür metodolojidir.

4.2 Özellikler

Scrum projeleri aşağıdaki özelliklere sahiptir [3];

- Esnek üretim vardır, teslim içeriği çevre tarafından belirlenir.
- Esnek bir zaman programı vardır, proje başlangıç planına göre erken veya geç tamamlanabilir.
- Küçük takımlar vardır. Projelerde birden fazla takım olabilir.
- Devamlı gözden geçirmeler vardır, takım ilerleyişi sık sık çevresel karmaşılıklar ve risk düşünülerek gözden geçirilir. (Genellikle 1 ile 4 hafta döngüsü)
- İşbirliği vardır, takım ve iş birimleri arasında işbirliği beklentisi vardır.
- Nesne tabanlıdır, her takım, açık davranış ve arayüzlerden oluşan nesneler ile çalışır.

Scrum yaklaşımı rekabet ortamı içerisinde yazılım şirketlerine kayda değer başarı yakalaması amacıyla kullanılır [3]. Endüstri analistleri Scrum'ın diğer yöntemlere göre nesne yönelimli teknikler ve araçlardan verim almak için daha uygun bir süreç olduğuna inanırlar [3].

Scrum düzenli geri bildirim ve kısa dönemli planlamalarla hedefe ulaşmayı sağlar. Yazılım geliştirme sürecinde ihtiyaç ve önem sıralamasına göre çalıştığından dolayı büyük taşları önceden yerleştirmeyi mümkün kılar. Günümüzde büyük taşlar zaman içerisinde müşteri ihtiyaçlarına göre şekillendiğinden müşterinin düzenli geri bildirimlerine göre yapılanmayı mümkün kılar.

Kontrol mekanizması öngörülemeyen koşulları yönetmeyi ve riski kontrol etmeyi amaçlar. Değişen durumlara karşı riski en az da tutmaya çalışır. Bunun sonucu süreç esneklik, duyarlılık ve güvenilirlik kazanır [28].

Scrum Teorisi

Deneysel süreç kontrol teorisi üzerine kurulmuş Scrum, riski kontrol etmek ve öngörebilmek için tekrarlanan artırımlı yaklaşımı sağlar. Üç temel ayak, her türlü deneysel kontrolde altyapıyı sağlar [28].

İlk ayak şeffalık:

Saydamlık, ürünü etkileyen süreci ve ürünü başarılı bir şekilde yönetmeyi garanti eder. Görünümler sadece saydam olmamalı aynı zamanda ne görünüyor iyi bilinmelidir. Bu durum, süreci denetleyen birinin birşeylerin yapıldığını, tanımlar ile yapılanın eşit olduğunu görmesine yarar [28].

İkinci ayak denetlemedir:

Sürecin çeşitli görünüşleri sık sık yeteri kadar denetlenmedir ki süreçte kabul edilemez uyuşmazlıklar bulunabilir. Denetlemenin sıklığı, bütünüyle işleme tabi tutan işin değişikliğine göre değişir. Muamma, denetleme sıklığı gerekliliğinin, sürecin denetleme toleransını aştığı zaman meydana gelir. Neyse ki bu yazılım geliştirmede çok olası değildir. Diğer etken, iş sonuçlarını denetleyenen insanların çalışkanlığı ve becerisidir [28].

Üçüncü ayak adaptasyondur:

Eğer denetleyici, sürecin bir veya daha fazla görünüşünün kabul edilen sınırların dışında olduğunu belirlerse, bu sonuç ürünün kabul edebilebilir olmayacağına işarettir, işte bu yüzden denetleyici işlenen süreci veya materyali denetlemelidir. Gerekli ayar veya düzeltmenin sonuç üründe fazla sapma olmaması için mümkün olduğu en kısa sürede yapılması gerekmektedir [28].

Scrum metodolojisinin diğer metodolijilere göre farkı (Şelale, Spiral, Yinelemeli gibi) Scrum analiz, tasarım ve geliştirme süreçlerini geliştirme süreci içerisinde öngörülemez olarak kabul eder ve değişikliklere her zaman açıktır. Esneklik, değişimlere cevap verebilirlik ve güvenilirlik sonuçlarıdır [3].

Diğer kriterlere göre karşılaştırma tablosu Çizelge 3.1'de aşağıdaki gibi verilmiştir;

Çizelge 3. 1 Scrum'ın diğer metodolojilerle karşılaştırma tablosu

Kriter	Şelale	Spiral	Yinelemeli	Scrum
Tanımlı süreçler	Gerekli	Gerekli	Gerekli	Planlama ve kapatma aşamasında
Final ürün	Planlamada tanımlanmış	Planlamada tanımlanmış	Proje esnasında şekilleniyor	Proje esnasında şekilleniyor
Proje maliyeti	Planlama esnasında hesaplanıyor	Kısmen değişken	Proje esnasında şekilleniyor	Proje esnasında şekilleniyor
Tamamlanma tarihi	Planlama esnasında belirleniyor	Kısmen değişken	Proje esnasında şekilleniyor	Proje esnasında şekilleniyor
Dış çevreye duyarlılık	Sadece planlamada	Planlama öncelikli	İterasyon sonlarında	Her zaman
Takım esnekliği,yaratıcılığı	Limitli,fazla yaratıcılık ve esneklik desteklenmiyor	Limitli,fazla yaratıcılık ve esneklik desteklenmiyor	Limitli,fazla yaratıcılık ve esneklik desteklenmiyor	İterasyonlar boyunca yaratıcılıkta ve esneklikte limit yok
Bilirkişi desteği(Harici)	Proje eğitimi esnasında	Proje eğitimi esnasında	Proje eğitimi esnasında	Gerekirse proje içinde takıma bile dahil edilebiliyor
Başarı olasılığı	Düşük	Orta düşüklükte	Orta Seviye	Yüksek

Scrum metodolojisinde yazılım ürün dağıtımları aşağıdaki değerler üzerinde planlanır [3];

- **Müşteri gereksinimleri:** Ürün çıktığı zaman hangi ihtiyaçları karşılayacak?
- **Zaman baskısı:** Ürün geliştirmesi için öngörülen zamandan daha erken bitirmenin ve dağıtımı yapmanın faydaları nelerdir?
- **Mücadele:**Rekabet ortamında başarılı olmak için gerekli gereksinimleri sağlamanın getirileri, öne çıkmanın maliyeti ve getiri kıyaslaması
- **Kalite:**Kalite gereksinimlerini sağlamak için gerekli olanlar nelerdir?

•**Vizyon:**Hangi değişiklikler gerekli proje hedefine yürüyebilmek için gereklidir?

•**Kaynak:**Personel ve diğer kaynak planlamaları

Bu değerler yazılım projesi için başlangıç planında belirlenir. Bununla birlikte bu değerler proje esnasında değişebilir. Başarılı bir geliştirme metodolojisi bu değişimlere ayak uydurabilen ve bu değişimleri başarı ile idare edebilecek yetenekte olandır [3].

Yazılım projelerinin en büyük kaynağı insandır. Sürecin nasıl çalıştığını anlatmadan önce bu süreçteki kişiler ve rollerden bahsetmemiz gerekir.

4.3 Scrum Rolleri

Scrum çevik yaklaşım gereği insanları, iletişimi ön planda tutan bir yaklaşımdır. Süreç içerisinde projede görevli kişilere yer verdiği gibi proje paydaşları ve diğer projeye katkısı olabilecek insanları sürece kabul ederek, aynı hedefte ilerleyen bir aile oluşturmayı hedefler. Çevik yaklaşımın anahtar özelliklerinden biri de bireyler ile müşteriyi sürecin parçası yapmak ve sürecin içinde ortak çalışmasını sağlamasıdır. Bu şekilde koşu sonrası değerlendirmeler ve artışların test edilmesi sonrası olacak geri dönüşler takımın başarısı için hayati önem taşır.

Scrum'da klasik "Proje Yöneticisi" rolü yoktur. Geleneksel proje yöneticilerinin görevleri Scrum'da üç rol paylaşır [33];

•Ürün Sahibi ürünü yönetir. (Yatırım geri dönüşünden sorumludur.)

•Scrum Master süreci yönetir.

•Takım kendini yönetir.

Takımın, Scrum Master ve Ürün Sahibi haricinde atanmış lideri yoktur, olmasına da gerek yoktur. Takım zaten kendi kendisini yöneteceği için bu tarz bir lidere ihtiyaç azalacaktır [33].

4.3.1 Ürün Sahibi

Ürün Sahibi projenin iş değeri açısından geri dönüşü ile sorumludur.

Proje içinde yatırımlar sonucu başarılı bir şekilde ürün alma konusunda müşteri temsil makamıdır. Projenin içindedir, müşteri tarafından görevlendirilmiştir, detayları takip eder, geri dönüşler verir.

Ürün Sahibi bir kişidir, bir komite değildir. Komiteler, Ürün Sahibi ile fikir alışverişi yapmak, öneride bulunmak amaçlı bulunabilir ancak ürün değişikliği yapmak isteyen insanlar Ürün Sahibi'ni ikna etmek zorundadır. Scrum'ı benimseyen şirketler öncelikleri ve gereksinimleri ayarlamak için kendi metodlarını zamanla geliştirirler [28].

Ürün Sahibi'nin başarılı olması için organizasyondaki herkesin onun kararlarına saygılı olması gerekir [28].

Hiç kimse takıma başka öncelikleri olan gereksinim listesi ile çalışmasını söyleyemez, takım da Ürün Sahibi'nden başka hiç kimseyi dinleyemez. Ürün Sahibi'nin içerikle ilgili kararları ve Ürün Gereksinim Listesi sıralaması şeffaftır, herkes tarafından görünür. Şeffaflık Ürün Sahibi'nin en iyiyi yapmasını gerektirir ve Ürün Sahibi'ni talep eden ve ödüllendirilen bir role büründürür [28].

Proje sorumlulukları bakımından, Ürün Sahibi'nin rolü, proje kapsamının, içeriğinin anlaşılması ve proje için pratik, pratik olmayan ve yaratıcı gereksinimler tanımlamak ve önceliklendirmektir.

Ürün Sahibi öncelik verilen fonksiyonelliğin gerçekleştirildiğini, işin ihtiyaçlarını karşıladığını ve iş açısından fayda kattığını garanti etmekle sorumludur.

Sürecin başarısı için Ürün Sahibi kendi organizasyonu için iş değeri katacak ürün gereksinimlerini tam olarak ifade etmelidir ki proje başarılı sonuçlanabilsin. Bu yüzden kritik bir rol üstlenmişlerdir, Ken H. Judy ve Ilio Krumins-Beens'in belirttiği gibi sürecin bir parçasıdırlar, proje başarısına doğrudan etkilidirler [34].

Ürün Sahibi'nin sorumlulukları;

- Paylaşımcı bir vizyonda çalışma
- Gereksinimleri toplama
- Öncelikleri yönetme ve önceliklendirme
- Her yineleme sonunda ürün kabülü yapma

- Dağıtım planını yönetme
- Projenin yatırım geri dönüşümünden sorumlu olma (ROI)

4.3.2 Scrum Master

Scrum Master, takımın Scrum değerlerine, pratiklerine ve kurallarına bağlı kalmasını garanti altına almakla sorumludur. Scrum Master, takımı ve organizasyonu Scrum'a adapte eder [28].

Bu rolü tarif etmenin birden fazla yolu vardır. Scrum Master bir ebeveyn gibi düşünülebilir; çünkü bir takımı oluşturduğunuz zaman başta bireyler kendilerini nasıl organize edeceklerini, nasıl çok görevli çalışacaklarını, Ürün Sahibi ile nasıl çalışacaklarını veya zaman kısıtlı olarak nasıl çalışacaklarını bilemeyebilirler. Bundan dolayı Scrum Master bir aile büyüğü gibi bunları takım üyeleri kendi başlarına yapabilene kadar öğretmek ve uygulatmak zorundadır. Benzer şekilde Scrum Master bir koç gibidir. Ekibin lideri, rehberi ve başardıklarında alkışlayanı olmalıdır. Aynı zamanda Scrum Master kuralları sıkı sıkı takip eden bir hakem gibi de sürecin doğru işlemesi için sorumluluklarını yerine getirir.

Scrum Master, Ürün Sahibi ile yakın çalışan işleri kolaylaştırıcı bir takım lideridir [35].

Scrum Master müşteri ve yöneticiler ile Ürün Sahibi'ni belirlemek ve desteklemek için beraber çalışır. Scrum Master Ürün Sahibi'ne işini nasıl yapacağını öğretir. Ürün Sahibi Scrum'dan istediği ürün için maksimum verimi almak için nasıl yöneteceğini bilmekle yükümlüdür. Eğer Ürün Sahibi bilmezse bu hususta Scrum Master sorumludur [28].

Scrum Master'ın sorumlulukları [35];

- Takımı yönlendirme, üretken olmasını sağlama
- Tüm rol ve fonksiyonlar arasında yakın iş birliği sağlama
- Engelleri kaldırma
- Takımı dış etkiler, baskılardan koruma
- Sürecin doğru işletilmesini sağlama

Scrum Master geliştirme takımının bir üyesi olabilir; mesela koşu görevleri gerçekleştiren bir geliştirici olabilir. Ama bu durum genellikle görevleri gerçekleştirmek ile sorunları çözmek için zaman çatışmasına neden olur ve Scrum Master asıl görevlerini yerine getiremeyebileceği için tavsiye edilmez. Değişmez kural ise Scrum Master ile Ürün Sahibi kesinlikle aynı kişi olmamalıdır [28].

4.3.3 Takım

Takım çapraz görevli çalışabilen bireylerden oluşur [35].

Scrum'da takımların özellikleri;

- Gereksinimlerin süre tahminini yaparlar [33]
- Küçük takımlardan oluşur (7+-2) [35]
- Koşuya başlarken hedefi seçip, çalışma sonucunu belirlerler
- Koşu hedefine ulaşmak için proje sınırları dahilinde herşeyi yapmakta serbesttirler
- Takım üyeleri kendi kendini organize eder
- Çalışma sonuçlarını ürün sahibine sunarlar ve geri dönüş alırlar

Takım üyeleri, geliştiriciler, testçiler, analistler, mimarlar, tasarımcılar ve hatta kullanıcılardan bile oluşabilir [33].

Takım çapraz görevlidir, bunun anlamı bu kişiler birbiri ile iş yaparken iş yapılan konuda iki tarafta bilgi birikimi sahibidir. Takım üyeleri arasında herhangi bir hiyerarşi yoktur [33]. Takım üyeleri üründe bir artırım yapmak için çeşitli yeteneklere sahip olmalıdır. Takım üyeleri genellikle özel yeteneklere de sahiptir. (Programlama, kalite kontrol, iş analizi, mimari, kullanıcı arayüz tasarımı veya veritabanı tasarımı gibi) Bununla birlikte takım üyelerinin paylaştığı beceriler arasından gereksinimi belirlemek ve kullanılabilir ürün için gerçekleştirme yapmak, diğerlerinden daha önemli bir yetenektir. Kodlamayı reddeden insanlar mimarlar veya tasarımcılar vb. bireyler Scrum takımları için uygun değildir. Herkes gerektiğinde yeni yetenekler öğrenilmesi veya eskilerin hatırlanması aktivitelerine katılır. Takımlarda ünvan yoktur ve bu kural istisnası olmayan bir kuraldır. Takımlar test, iş analizi işleri gibi alt gruplara da sahip değildir [28].

Kendi Kendini Organize Etme (Self Organization)

Kendi kendini organize etme yaklaşımı hiçbir şekilde kişilere müdahele etmeme, başıboşluk ilkesini ima etmez, aksine bu tarz takımlar yüksek disiplinli takımlardır [33].

Takıma teslime uygun taahhütleri ile tam özerklik ve daha fazla sorumluluk verilir. Takım makul riskleri almak için cesaretlendirilir, başarısızlık karşısında kendi kendine çözüm alabilme yeteneği beklenir [33].

Yüksek güven ve yüksek vaat kendi kendine organize olan takımların beklenen sonucudur [33].

Scrum'daki yeni takımlar sınırları görmek ve sahiplenme bilincini almak için başlarda teşvik edilmek isteyeceklerdir. Bu yüzden eski alışkanlıkları, klasik yöntemlerden gelen hafızalarındaki gelenekleri aşmaları gerekir [33].

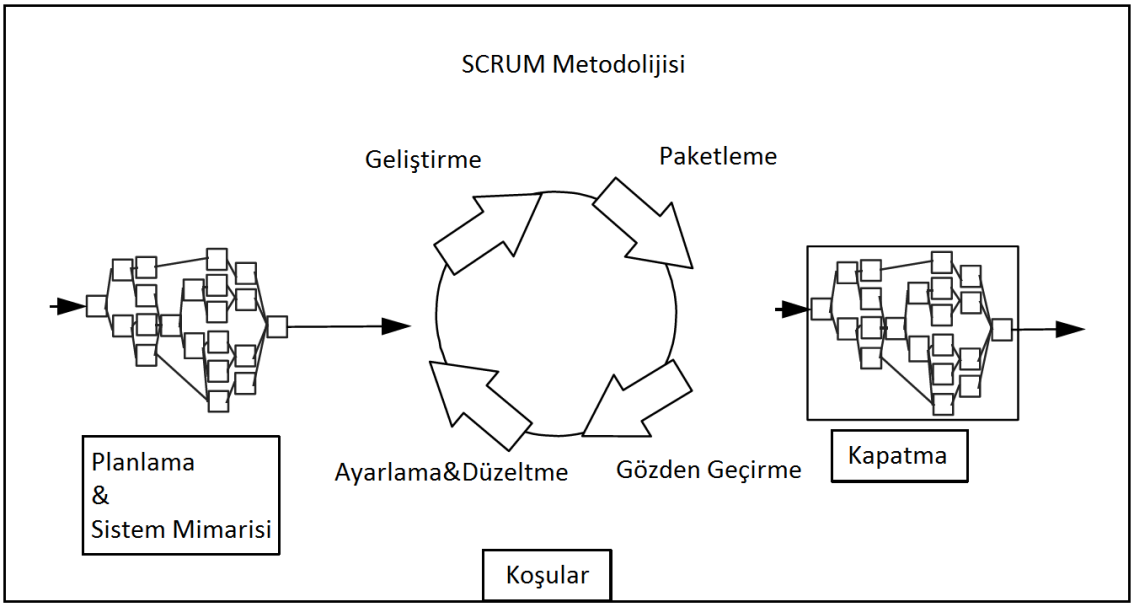
Kendi kendine organize olmak Scrum için bir seçenek değil, bir temel ilkedir. Bu olmadan yüksek performanslı takımlar olmayacaktır [33].

4.4 Scrum'ın Aşamaları

Scrum başlangıç ürün gereksinimlerinin oluşturulup, kullanılacak mimari, teknik detaylar, sözleşmeler vb. belirlendiği hazırlık evresiyle başlayıp, yinelemeli koşularla yazılımın geliştirildiği, ara dağıtımlar ile müşteriye ürünün parça parça sunulduğu ve final koşusunun ardından testler ve dökümantasyon ile müşteriye istediği ürünün sunulduğu bir proje yönetim metodolojisidir.

Scrum'da organizasyonlar Şekil 3.2'de olduğu gibi başlangıç ve bitiş aşamaları arasında iteratif ve artırımlı olarak geliştirmeyi gerçekleştirir.

Başlangıç ve bitiş aşamaları (planlama ve kapatma) tanımlanmış süreçlerden meydana gelir. Tüm süreç adımları ve giriş çıkışlar tanımlıdır. Bazı yinelemelerle birlikte planlama aşamasındaki akış doğrusaldır [3].



Şekil 3. 2 Scrum aşamaları

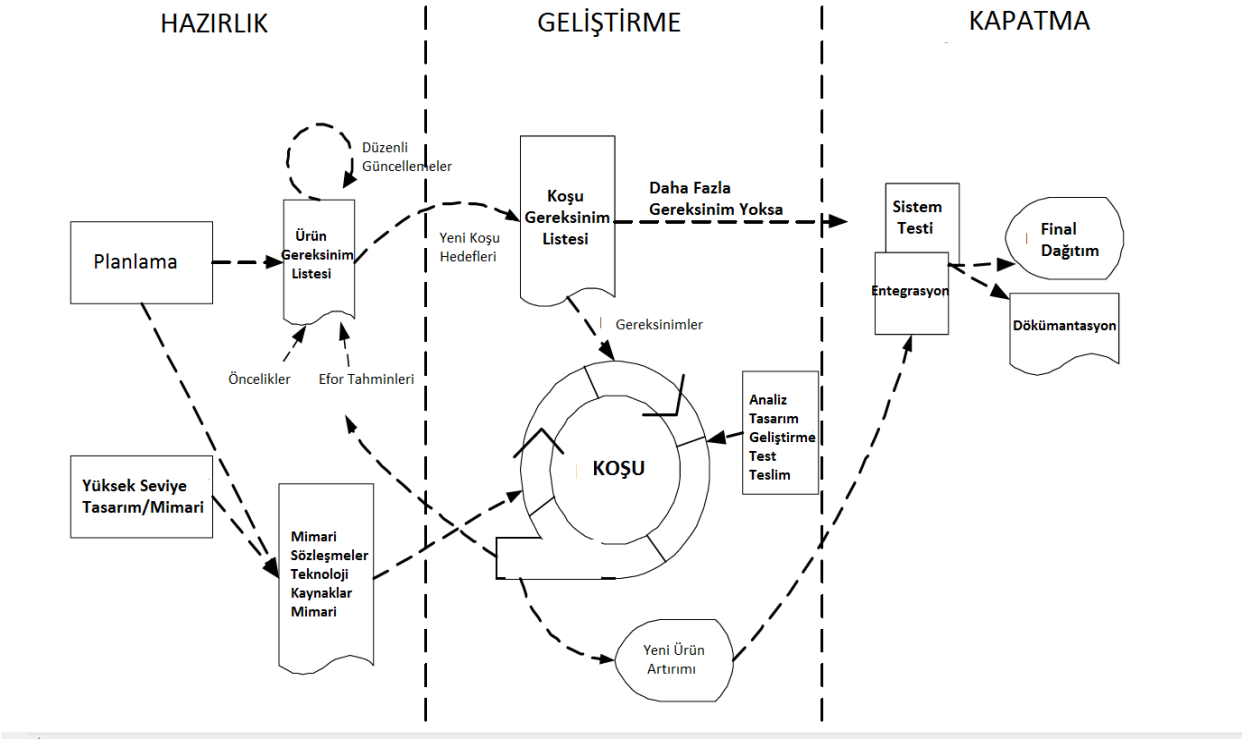
Koşu aşaması deneyseldir. Koşu aşamasındaki süreçler tanımlanmamış veya kontrolsüzdür. Bu nedenle kara kutu yaklaşımı gibi dışarıdan bir kontrole ihtiyaç duyar. Risk yönetimini barındıran kontrol her yinelemede riski en aza indirmeyi, değişikliklere cevap vermeyi ve maksimum esnekliği sağlar [3].

Koşular doğrusal değil ve esnektir. Koşular nihai ürünü geliştirmek için kullanılır [3].

Proje kapatma aşamasına kadar açık formda devam eder. Sunulan ürün planlama veya koşu aşamalarında değişiklik gösterebilir [3]. Ürün, ortama dayalı proje esnasında belirlenir [3]. Scrum takımlara yapılacak işin miktarını ve en iyi nasıl yapılacağına dair seçim şansı verir [35].

Geliştirme aşamasında müşteriden gelebilecek değişiklik taleplerine adaptif olarak geliştirilen süreç, süreç içerisinde belirli dönemlerle müşteri taleplerine göre gereksinimleri önceliklendirir. Bunu yaparak müşteriye ürün artırımlarında daha az değeri olan gereksinimler yerine istediği gereksinimlerin çıkmasını sağlayarak istediği ürünü görmesini ve müşteri tatminini sağlar [35].

Scrum'da aşamaların detaylı görünümü Şekil 3-3'deki gibidir;



Şekil 3. 3 Scrum'da süreçlerin genel görünümü [36]

4.4.1 Hazırlık

Planlama: Geliştirilecek ürünün maliyet ve süre belirlemelerini de yaparak gereksinim listesinin çıkarılması aşamasıdır. Eğer yeni bir sistem geliştirilecekse bu evre de kavramsal yorum ve analiz bulunur. Eski bir sistem geliştirilecekse de evre sadece sınırlı analizden oluşur [3].

Süreç adımları aşağıdakilerden oluşur;

- Geniş kapsamlı gereksinim listesinin çıkarılması
- Yapılacak dağıtımların (bir veya daha fazla) çıkış tarihinin ve fonksiyonel özelliklerinin belirlenmesi
- Başarılı geliştirme için uygun dağıtım gereksinimlerinin belirlenmesi
- Dağıtımlar için gereksinim eşleştirilmenin yapılması (Hangi gereksinim hangi dağıtımda gerçekleştirilecek?)
- Dağıtımlar için takımların belirlenmesi
- Risk değerlendirmesi ve risk kontrollerinin belirlenmesi

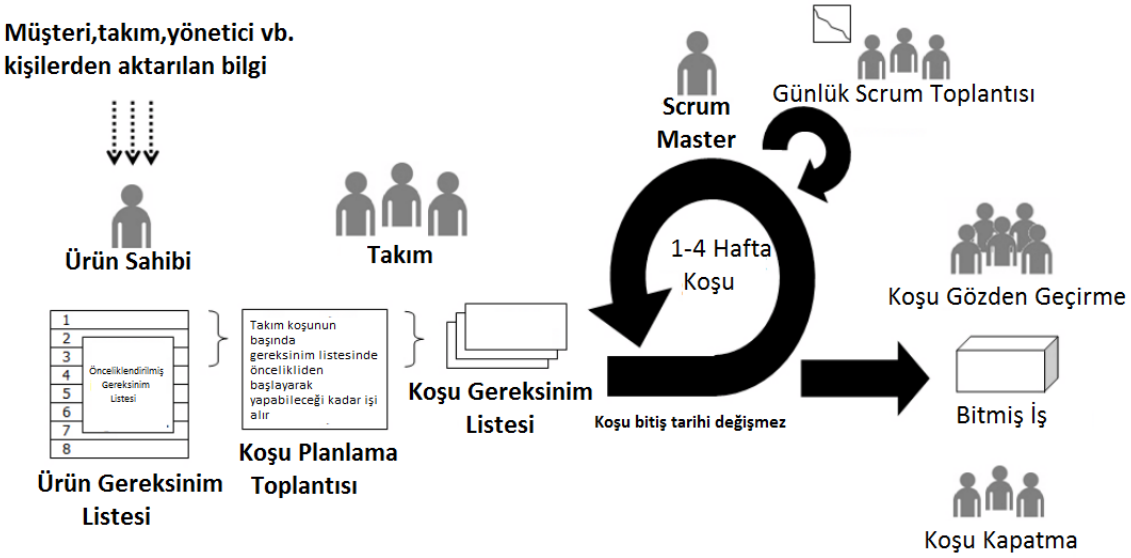
- Gözden geçirmeler ve olası gereksinim değişikliklerinin belirlenmesi
- Geliştirme araçları ve altyapının onaylanması
- Dağıtım maliyeti, geliştirme, materyal toplama ve pazarlama maliyetlerinin hesaplanması
- Yönetimi, desteklerin gözden geçirilmesi ve onaylanması

Mimari ve Yüksek Seviye Tasarım: Bu evrede gereksinim listesi elemanları nasıl gerçekleştirilecek bu belirlenir. Yüksek seviye tasarım ve sistem mimarisi modifikasyonu içerir [3].

- Onaylanmış gereksinim listesinin gözden geçirilmesi
- Önem sırasına göre gereksinimlerin derecelendirilmesi
- Alan analizi
- Sistem tasarımının yeni gereksinimleri kabul edebilecek şekilde yeniden düzenlenmesi
- Geliştirme veya değişiklikler ile ilgili sorunlar ve problemlerin tanımlanması
- Gözden geçirme toplantılarının, görevlerin ve her bir gereksinimin yeterlilik şartlarının tanımlanması

4.4.2 Geliştirme

Geliştirme aşaması Şekil 3.4'de görüldüğü gibi yinelemeli geliştirme adımlarından oluşur. Yönetim süre, rekabet, kalite, fonksiyonalite, tamamlanan adımlar ve kapatma safhası ile ilgilenir. Bu yaklaşım eşzamanlı mühendislik olarak da bilinir [3].



Şekil 3. 4 Scrum metodolojisi geliştirme evresi [35]

Dağıtım Planlama Toplantısı

Dağıtım planlamasının amacı, takımın ve diğer bağlı organizasyonların plan ve hedeflerinin belirlenmesidir. Dağıtım planı şu soruları cevaplar [28];

•Mümkün olan en iyi yolla ürün elde etmeyi nasıl gerçekleştirebiliriz?

•Nasıl artan müşteri isteklerini karşılayabiliriz?

•Nasıl müşteri memnuniyeti sağlayabilir, dönüş veya artırım yapabiliriz?

Dağıtım planı dağıtımın sahip olacağı hedefini, ürün gereksinimindeki yüksek öncelikleri, büyük riskleri, genel özellikler ve fonksiyonaliteyi içerir. Ayrıca hiç bir değişiklik olmaması halinde tahmini çıkış tarihi ve maliyet de belirlenir.

Organizasyon daha sonra ilerlemeyi denetleyebilir, bu dağıtım planı üzerinde koşudan koşuya esasında değişiklik yapabilir [28].

Dağıtım planlaması tamamen opsiyoneldir. Eğer Scrum takımı toplantı olmadan çalışmaya başlarsa, araçların eksikliği ileride çözülmeye ihtiyaç duyan bir engel olarak açık kalacaktır. Engelleri çözmek, daha sonra Ürün Gereksinim Listesi'nin bir maddesi olur. Ürünler Scrum'da iteratif olarak üretilir, her koşu en değerli ve riskliden başlayarak üründe bir artırım gerçekleştirir. Daha, daha fazla koşular, ürüne ek artırımlar kazandırır. Her artış, tam ürünün potansiyel olarak dağıtım yapılabilir bir

parçasıdır. Ürün değeri açısından yeteri kadar artırımlar sağlandığı zaman, yatırımcıların kullanımı için ürün versiyonu çıkarılır [28].

Çoğu organizasyon zaten dağıtım planlama sürecine sahiptir, bu süreçlerin çoğunda planlamanın çoğu dağıtımın başında yapılır ve zamanla değişime bırakılır. Scrum dağıtım planında genel hedef ve olası sonuçlar tanımlıdır. Bu dağıtım planı genellikle genellikle geleneksel dağıtım planlarının süresinin %15, %20'sinden fazla gerektirmez. Bununla birlikte Scrum dağıtımı her Koşu Gözden Geçirme, Koşu Planlama Toplantısı'nda planlama yapılır, ayrıca her günlük toplantıda da planlama işlemi yapılır. Genelde Scrum dağıtım planlama eforu azda olsa geleneksel dağıtım eforlarından daha fazla efor gerektirir [28].

Geliştirme Koşuları

Zaman, gereksinimler, kalite, maliyet ve rekabete bağlı kalarak gereksinim listesindeki özellikleri geliştirerek ürün artırımı yapmak amacı ile gerçekleştirilen işlemlerdir. Bunlar çok sayıda iteratif ve tüm gereksinimler bitene kadar devam eden sürekli koşulardır [3].

Geliştirme süreci aşağıdaki temel adımlardan oluşur;

- Takım ile dağıtım planlarının gözden geçirme toplantısının yapılması
- Ürün standartların gözden geçirilmesi, dağıtımı ve düzenlenmesi
- Ürün dağıtım yapılmaya hazır hale gelene kadar yapılan iteratif koşular ile ürün geliştirilmesi

Koşu 1 ile 4 hafta arası önceden tanımlanmış aktivite setlerinden oluşur. Süre ürün karmaşıklığı, risk değerlendirmesi vb. nedenlere göre değişiklik göstermektedir. Koşu hızı ve yoğunluğu seçilen yöntemlere göre değişiklik gösterir [3].

Koşu bir yinelemedir. Koşular zaman aralıklarıdır. Koşu esnasında Scrum Master koşu hedefini etkileyecek bir değişiklik yapılmasını engellemeyi garanti etmekle yükümlüdür. Hem takım kompozisyonu hem de kalite hedefleri koşu boyunca korunmalıdır. Koşu, Koşu Planlama Toplantısı, geliştirme işi, Koşu Gözden Geçirmesi ve Koşu Kapatma Toplantısı'ndan meydana gelir. Koşu bir önceki koşudan hemen sonra başlar ve koşular arası zaman yoktur [28].

Her proje zaman diliminde planın iyi olup olmadığını gösteren bir ufka sahiptir. Eğer ufuk çizgisi çok uzun olursa tanımlar değişebilir, çok fazla değişken girebilir, risk çok büyük olabilir. Scrum bir proje için bir altyapıdır ve ufuk bir aydan fazla olmamalıdır. Yeterli karmaşıklıkta uzun ufuklar çok risklidir. Projenin tahmin edilebilirliği en az bir ayda kontrol edilmelidir, bu işlem haricinde proje kontrolden çıkabilir veya tahmin edilemez olabilir [28].

Koşular, koşu süresi tamamlanmadan iptal edilebilir. Buna sadece her ne kadar Scrum Master, ortaklar, takım tarafından etkilensede Ürün Sahibi'nin yetkisi vardır [28].

Scrum bireylerin uzun soluklu hedefler yerine daha kısa, bütünün parçalanması ile oluşan hedeflere yönlendirerek motive olmasını sağlar.

Koşu Scrum döngüsünün kalp atışıdır. Koşular Koşu Planlama Toplantısı ile başlayarak, Koşu Gözden Geçirme ve Koşu Kapatma Toplantı'ları ile sonlanır. Koşuların uzunluğu sabittir ve asla uzatılmaz. Günümüzde çoğu takım koşu süresi olarak 3, 4 hafta seçiyor [33].

Koşular, Koşu Planlama Toplantısı ile başlar;

Koşu Planlama Toplantısı

Her koşu başlangıcında Koşu Planlama Toplantısı gerçekleştirilir [35].

Koşu planlama toplantısının ilk kısmında Ürün Sahibi ve takım Ürün Gereksinim Listesi'ni gözden geçirir, gereksinim elemanlarının hedeflerini, içeriklerini belirler, takımın Ürün Sahibi'nin düşüncüleri ile aynı paralelde olması sağlanır [35].

Toplantının ikinci kısmında, takım üyeleri Ürün Gereksinim Listesi'nden en üstten (Ürün Sahibi tarafından kritik, iş değeri katan gereksinimlere yüksek öncelikler verilir) başlayarak koşu sonucunda gerçekleştirilecek gereksinimlerden oluşan Koşu Gereksinim Listesi'ni oluştururlar. Bu Scrum'ın anahtar uygulamasıdır, takım Ürün Sahibi tarafından belirtilen önceliklendirilmiş gereksinimlerden ne kadar yapacağını belirler ve taahhüt eder [35].

Bu durum daha güvenli bir taahhüt sunar çünkü takım ne kadar gerçekleştirim yapacağı ile ilgili kendi kararını verir, aksi takdirde ikinci bir kişi tarafından verilecek kararda o iş için ne kadar çalışma gerektiği bilgisi gibi bilgiler daha az gerçekçilik taşıyabilir [35].

Ürün Sahibi takımın ne kadar gereksinim gerçekleştireceği hakkında herhangi bir kontrole yetki sahibi değilse de bilir ki takım Ürün Sahibi'nin önceliklendirdiği gereksinim elemanlarından en önemlilerinden başlayarak Koşu Gereksinim Listesi'ni oluşturacaktır. Takım eğer mantıklı ise aşağıdan daha çok gereksinim elemanı alma ve Koşu Gereksinim Listesi'ne ekleme hakkı vardır [35]. (Örnek verilecek olursa bazen daha az öncelikli elemanı almak daha öncelikli bir elemanın gerçekleştirilmesinin hızlanmasına yardımcı olabileceği durumlarda bu işlem gerçekleştirilebilir.)

Planlama toplantısının aktiviteleri;

- Ürün Sahibi, Ürün Gereksinim Listesi'ni yeniden önceliklendirir
- Ürün sahibi, Scrum Master ve takım sonraki koşuda tamamlanabilecek işi kararlaştırır
- İşler öncelik sırasına göre en öncelikliden başlanarak seçilir
- Ürün Sahibi diğer müşteri bireyleri arasında tam anlaşmayı sağlamak ile yükümlüdür, bu şekilde takım tek bir hedefe odaklanır
- Takım ve Ürün Sahibi koşunun hedefini belirler
- Takım yapabileceği işi seçmesi için beklenir
- Seçilen görevler Koşu Gereksinim Listesi'ne aktarılır
- Koşularda önceki performans değerlerinden geri besleme alınarak analiz yapılır
- Eğer Koşu Planlama Toplantısı'ndan sonra %20'den fazla çalışmayı gerektirdiği görülürse daha iyi planlaması gerektiği anlaşılır ve koşu sonlandırılması esnasında bu durum tartışılır

Koşu esnasında takım hergün Scrum Günlük Toplantısı'nda bir araya gelir;

Günlük Scrum Toplantısı

Koşu başladıktan sonra takım sürecin başka bir anahtar aktivitesi olan Günlük Scrum Toplantısı'nı gerçekleştirir. Bu kısa toplantı (onbeş dakika) her iş gününde genelde belirlenen saatte gerçekleştirilir ve tüm takım katılır [35].

Takımın ilerleyişini ve karşılaştıkları engelleri görmek için önemli bir fırsattır. Teker teker tüm ekip üyeleri şu soruların cevaplarını verir [35];

- Dün ne yaptım?
- Bugün ne yapacağım?
- Önümde olan engeller ve karşılaştığım sorunlar neler?

Scrum Master toplantı esnasında notlar tutar ve sorun yaşayan üyelere toplantıdan sonra yardımcı olur [35].

Günlük Scrum Toplantısı kesinlikle bir tartışma platformu değil, işlerin ne durumda olduğu ve varsa sorunların paylaşıldığı bir toplantıdır, eğer tartışma gerekiyorsa bu toplantıdan sonra gerçekleştirilir [35].

Ürün Sahibi, Scrum Master ve diğer paydaşlar toplantıya katılabilir ama bu kişiler toplantı bitmeden soru sormamaya ve tartışma açmamaya dikkat etmelidir. Herkes bilmelidir ki bu toplantının amacı ekibin kendi kendine rapor vermesidir, ne Ürün Sahibi'ne, ne Scrum Master'a ne de paydaşlara rapor vermek değildir [35].

Standart dışı olsada bazı takımlar Ürün Sahibi'nin toplantıda bulunmasını ve çalışmalar ile ilgili görüş belirtmesini faydalı bulurlar [35].

Bu toplantılar çok faydalıdır ve bazı sonuçları aşağıdaki gibidir:

- Engeller oluştu ise yönetim tarafından ortadan kaldırılır
- Daha az gereksiz tekrarlanmış efor harcanır
- Takım üyeleri arasında daha iyi birliktelik ve uyum sağlanır
- Hedef netleşir ve takım tarafından kabul edilir
- Takım iletişimini sağlar
- Takımın önünde yaptıklarını açıklayacak olmak bireyi başarılı olma yönünde teşvik eder
- Takım dinamiği ve etiği inşa edilir

Bu toplantılar için belirlenmiş bazı kurallar vardır;

- Her çalışma gününde toplantılar aynı saatte ve aynı yerde olmalıdır. Zaman olarak günün ilk işi olarak bu toplantıyı gerçekleştirmek idealdir.

- Tüm takım üyelerinin katılımı zorunludur. Eğer mazeret sonucu katılamayan birey varsa durumunu bir başkasına bildirir.
- Scrum Master, kararlaştırılmış zamanda kim olursa olsun toplantıya başlar. Geç kalan herhangi bir üye organizasyon içi yöntemlerle cezalandırılır.
- Scrum Master toplantıya solundaki kişi ile başlar. Saat yönünde tüm kişiler bitene kadar herkes konuşur.
- Takım üyeleri, takıma hitap etmelidir. Bu, bir "Scrum Master'a sunum" toplantısı değildir.
- Toplantı esnasında sadece bir kişi konuşur. Diğer katılımcılar dinlemek zorundadır.

Toplantıdan sonra takım üyeleri her bir görev için Koşu Gereksinim Listesi'nde efor güncellemesi yaparlar. Bu şekilde takım gün ve gün ilerleyişini Koşu Azalan Zaman Grafiği'nde görebilir [35].

Takım, ürünü koşu sonrası müşteriye ve diğer paydaşlara Koşu Gözden Geçirme Toplantısı'nda sunar;

Koşu Gözden Geçirme Toplantısı

Koşunun bitiminde, takımın koşu esnasında ne geliştirdiğinin gösterilmesi amacıyla örnek bir gösterim yaptığı Koşu Gözden Geçirme Toplantısı yapılır. Bu toplantı da Ürün Sahibi, takım üyeleri, Scrum Master, müşteri, paydaşlar, uzmanlar, kullanıcılar gibi diğer ilgili insanlar bulunabilir [35].

Koşu Gözden Geçirme Toplantısı'nın odak noktası takımın geliştirdiği üründür [33].

Bu toplantı takımın verdiği bir sunum değildir. Herhangi bir sunu gerekli değildir ve takımın bu toplantıya hazırlanması bir saati geçmemelidir. Bu sadece uygulamanın gösterildiği yapıldığı bir toplantıdır ve katılan herkes fikir bildirme ve soru sorma konusunda serbesttir [35].

Toplantının kuralları;

- Takım, Koşu Gözden Geçirme hazırlığını 1 saatten fazla uzatmamalıdır
- Gerçekleştirilemeyen özellik sunulamaz

- Ürün kendi üretim ortamında takımın en güvenli hissettiği ve kendi üretim donanımında sergilenmelidir
- Koşu Gözden Geçirmesi takım üyesinin koşu hedefini sunması ile başlar ve biten gereksinimlerin gösterilmesi ile devam eder. Diğer takım üyeleri koşu esnasında neyin doğru neyin yanlış gittiğini tartışabilir.
- Koşu Gözden Geçirmesi'nde zamanın çoğu ürünü sunan takım üyelerine harcanır ve daha sonra var ise müşteri soruları cevaplanır ve talepler not alınır.
- Sunumun sonunda müşteri izlenimleri, var ise değişiklik talepleri ve bunların öncelikleri alınır.
- Ürün Sahibi temsilcisi olduğu müşteriler ile ürünü tartışır ve Ürün Gereksinim Listesi'nin yeni halini ve öncelik ayarlamalarını yapar.
- Müşteriler yorum yapmakta özgürdür, gözlemlerini, eleştirilerini ve takdirlerini paylaşabilirler.
- Müşteriler teslim edilmemiş özellikleri öncelik ayarlaması yaparak bir sonraki koşuda gerçekleştirilmesi için Ürün Gereksinim Listesi'ni güncelleyebilir.
- Müşteriler sunumdan sonra Ürün Gereksinim Listesi'ne yeni eklemeler yapabilirler.
- Scrum Master toplantı kurallarını ve katılımları denetlemekle yükümlüdür.
- Toplantının sonunda bir sonraki Koşu Gözden Geçirme Toplantısı'nın yeri ve zamanını bildirilir.

Takım kendi koşu değerlendirmesini Koşu Kapatma Toplantısı ile yapar;

Koşu Kapatma Toplantısı

Koşu Gözden Geçirme Toplantısı'nı takiben takım Koşu Kapatma Toplantısı için bir araya gelir. Bu günümüzde uygulamada bazı takımların es geçtiği bir pratiktir ancak bu Scrum'ı başarılı kılan etkenlerden birisidir. Bu toplantı takımlar için neyin çalışıp neyin çalışmadığı ve denenecek değişiklikler ile ilgili tartışmak için fırsat yaratır. Takım, Ürün Sahibi ve Scrum Master toplantıya katılır. Yabancı katılımcılar toplantıya katılım sağlayabilir bazen de birden fazla Scrum takımlarının olduğu organizasyonlarda takımların birbirinin kapatma toplantılarına katılması iyi bir yaklaşımdır [35].

Koşu Gözden Geçirmesi ne olduğu ile ilgilenirken, Koşu Bitirme Toplantısı nasıl ile ilgilenir.

Bu toplantıda koşunun ilerlemesine etki eden her şey konu olabilir;Süreçler, pratikler, iletişim, çevre, araçlar...

Scrum devamlı geri beslemeler ile kendini devamlı geliştiren koşulardan oluşan bir süreç yapısıdır. Bu toplantı bu özelliğin hayata geçirildiği en önemli ortamlardan birisidir.

Tüm takım elemanlarının dürüstçe kendini ifade edebileceği bir toplantıdır. Bu nedenle Scrum Master'ın takımı serbest bırakması ve tarafsız olması gerekmektedir, aynı şekilde takım üyeleri gerekirse müşteri ile ilgili şikayetlerini de getirebilir.

Bu toplantının anahtar maddeleri;

- Süreç geliştirmeleri her koşu sonunda yapılır ve takım her koşudan sonra süreci geliştirmeyi garanti eder
- Koşu Kapatma Toplantısı tüm takım üyeleri arasında bir işbirliği toplantısıdır
- Tüm takım üyeleri daha iyinin nasıl olacağı konusunda fikir yürütürler
- Takım en çok karşılaşılan problemler konusunda (özgüven eksikliği ve kendini yönetme) kendini geliştirir.
- Scrum Master aldığı geri beslemeler ile aksiyonlara öncelik verir
- Takım halinde yapılan tartışmalar ilerlemenin ve başarılı olmanın en önemli etkenidir

Yeni Koşuya Başlama

Koşu Gözden Geçirme Toplantısı'nın ardından, Ürün Sahibi koşudan sonra gerçekleşen güncellemelerle ilgili tüm giriş bilgisini alır, Ürün Gereksinim Listesi ile birleştirir, yeni elemanlar ekler, eskileri günceller, siler, yeniden önceliklendirir [35].

Bu güncellemeler tamamlanıp tamamlanmaz çevrim yeni Scrum Planlama Toplantısı ile yeniden başlatılır [35].

Hangi durumlarda koşu iptal edilme ihtiyacı olur?

Yönetim koşu hedefi artık geçersiz, amaçsız, güncel olmadığı zaman iptal kararı alabilir. Şirket yönelim değişiklikleri, pazar veya teknoloji koşulları bunları tetikler. Genelde koşu koşulların anlam sağlamadığı hissedildiği zaman iptal edilmelidir. Bu duruma koşuların kısa sürelerinden dolayı azda olsa karşılaşılmaktadır.

Bir koşu iptal edildiği zaman biten ürün gereksinimleri gözden geçirilir. Eğer sunulabilir artış olduğu görünürse kabul edilir. Diğer ürün gereksinimleri tekrar Ürün Gereksinim Listesi'ne başlangıç tahminleri ile geri konur. Yapılan çalışmalar kayıp olarak geçer. Koşu iptalleri takım tekrar toplanıp diğer koşuyu başlatmak için planlama aşamasına geçene kadar kaynakları tüketir [28].

Koşu iptalleri genel de takım için travmatikdir ve az rastlanır [28].

4.4.3 Kapatma

Projeyi Kapatma: Dağıtım için hazırlık; tamamlanma dökümantasyonu, test ve dağıtımdan oluşur.

4.5 Scrum Araçları

Scrum sürecin ilerlemesi ve başarısı için bazı doküman ve araçlar kullanır.

4.5.1 Ürün Gereksinim Listesi

Ürün Gereksinim Listesi, proje boyunca yapılması gereken iş elemanlarının basit bir listesidir. Bu listeye herhangi biri tarafından ekleme yapılabilir ama ancak ve ancak Ürün Sahibi, takım tarafından gerçekleştirilecek gereksinimlerin önceliklendirmesini yapabilir. Elbette bunu paydaşlar ve takımla tartışarak yapmalıdır [33].

Gereksinimler acildir, bunun anlamı biz ürün içinde ne isteniyor önceden tüm detayları bilemeyiz. Dolayısıyla Ürün Gereksinim Listesi canlı bir dökümandır, geçerli ve kullanışlı olması için devamlı bakım gerekir. Birçok yeni eleman zamanla eklenecek, birçok eleman daha küçük parçalara ayrıştılacak, bazı gereksinimlerin artık gereksiz veya anlamsız olduğu keşfedilip silinecektir. Ayrıca gereksinimler değer, zaman ve maliyet kriterleri göz önüne alınarak boyutlandırılmalıdır. Ürün Gereksinim Listesi'ndeki sıralamalar gün ve gün değişiklik gösterebilecektir [33].

Ürün Gereksinim Listesi genellikle kullanıcı hikayelerinden oluşur ve kullanıcı bakış açısından bakılır [33].

Tahminin kesinliğini ve önceliklerin doğru verilmesi projenin başarısı için önemli faktörlerden biridir. En kapsamlı özellik isteğinden, en küçük, pratik olmayan isteğe kadar herhangi bir fikir veya istek, Ürün Gereksinim Listesi'ne eklenebilir.

Bu liste Ürün Sahibi ve müşteriler için kapsamı tarif etmek, ilerleyişi görmek ve gereksinimlerini tam olarak ifade etmek için başarılı bir araçtır.

Birden fazla Scrum takımı, aynı ürün üzerinde çalışabilir. Bir Ürün Gereksinim Listesi üründe yaklaşan işi tanımlamak için kullanılır. Ürün Gereksinim Listesi kullanılan grup elemanlarını tanımlar. Gruplama özellik seti, teknolojii veya mimariden meydana gelebilir ve genel de Scrum takımının çalışmasını organize etmek için kullanılır [28].

Scrum takımları genellikle koşunun %10'nunu Ürün Gereksinim Listesi'ni düzenlemek için kullanılır. Taneciklilığin düzenini gerçekleştirirken, Ürün Gereksinim Listesi'nin en üstündekiler bir koşu içinde kullanılmak için ayrıştırılır. Düzenleme işleminden önce bunlar analiz edilir ve görüşülür. Koşu Planlama Toplantısı yapılırken, Ürün Gereksinim Listesi tanımlanmış ve sıralanmıştır bu sayede özellikleri Koşu Gereksinim Listesi için kolayca seçilir [28].

4.5.2 Koşu Gereksinim Listesi

Koşu Gereksinim Listesi, bir görev çizelgesi gibi geçerli takımın koşu da yapmayı taahhüt ettiği gereksinimlerin fiziksel sunumudur [33].

Ekibe ve diğer insanlara, planladıkları işleri, koşuyu ve durumunu göstermeye yarar [33].

Koşu Gereksinim Listesi'nin özellikleri;

- Ürün Gereksinim Dökümanı'ndan ürünün fonksiyonelliğine eklenecek özellikleri içerir
- Koşu Gereksinim Dökümanı'nda ki özelliklerin Ürün Gereksinim Dökümanı'nda bulunması gerekir
- Görevlere saat tahmini yapılır(1-16)

- 16 saatten fazla süreceği tahmin edilen görevler küçük görevlere ayrıştırılır
- Takım üyeleri görevleri üstüne alır
- Herhangi bir takım üyesi, Koşu Gereksinim Listesi'ndeki görevlerini ekleyebilir, silebilir, değiştirebilir
- Eğer iş net olarak tanımlanamıyorsa daha büyük bir süre tahmini ile parçalara ayrılır
- Koşu boyunca üzerinde çalışılan işler için süre tahminleri güncellenir
- Takımın herhangi bir üyesi Koşu Gereksinim Listesi'ne ekleme veya çıkarma yapabilir

Takımın işleri yoluna koyma adına idare ile ilgili çok az görevi vardır. Koşu Gereksinim Listesi'ni güncel tutmak bunlardan biridir. Scrum deneysel bir süreç şeklinde ilerler ve sürecin başarısı için koşularda yaşanılanların geri besleme olarak ilerdeki koşulara aktarılması gerekir. Koşu Azalan Zaman Grafikleri(Sprint Burndown Chart) eğer Koşu Gereksinim Listesi devamlı güncel tutulmuyorsa ve bu grafikler takımın başarı durumunu göstermiyorsa yararsız olacaklardır.

- Tahmini süreler günlük olarak Scrum toplantılarından sonra güncellenir.
- Bu kişisel zaman takibi değildir. Scrum içinde takımın çalıştığı zamanın miktarını izlemek için düzenlenmiş bir mekanizma yoktur. Sadece kalan zaman grafikleri ile performans takip edilir. Organizasyonlar kendi içlerinde bunu gerçekleştirebilir.
- Takım hedefleri karşılamak ile ilgilenir. Hedefi ne kadar sürede karşıladığı ile değil. Scrum harcanan efor ile değil netice ile ilgilenir.

Takımın, bir vazifeyi yapıldı kabul ettiği kriterlerin tüm koşularda tutarlı olarak uygulaması önemlidir.

Önemli olan takımın vazifelerin bitme durumları ile ilgili müşteriye karşı dürüst olmalarıdır.

Bir vazifenin tamamlanıp tamamlanmadığı ile ilgili aşağıda ki hususlar önemlidir;

- Kodun standartlara uygun olması
- Kodun anlamlı okunabilir olması
- Birim testleri geçmesi

- Entregrasyon testinin başarılı olması.
- Kaynak kodun kontrolü
- Kalite kabul testinden geçmesi

Çevik süreçlerin çevrimsel yapısı gereği koşu görevlerinin mühendislik yöntemleri ve araçlar ile otomatik kontrol mekanizmasının, kurallar yapısının iyi tanımlanması ve uygulanması gerekir. Bunların dışındaki durumlarda ürün artışlarındaki ürün kalitesi garanti edilemez.

Koşu boyunca gerçekleştirilecek Koşu Gereksinim Listesi elemanları sabittir.

Ancak Koşu Gereksinim Listesi bazı nedenlerden dolayı değişebilir;

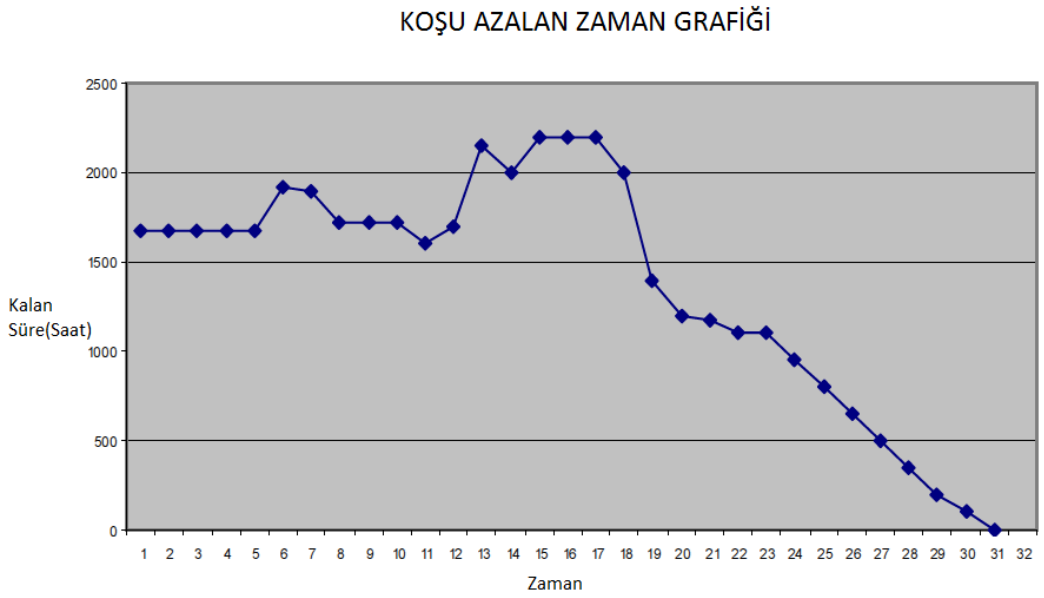
- Geliştirme ekibi, koşu içerisinde zaman ilerledikçe işi daha iyi kavrayabilir ve bazı zamanlar koşuyu tamamlamak için bazı Ürün Gereksinim Listesi elemanlarından ekleme yapma gereksinimi duyabilir
- Hatalar, koşu esnasında tespit edilip, ekstra görev olarak tanımlanabilir. Bunlar bitmemiş görevlerin öncelikli işleri olduğu için ekstradan bunların çözümü için değişiklikler yapılabilir.
- Ürün sahibi, koşu hedefini takıma daha iyi anlatabilme adına takımla birlikte çalışabilir. Scrum Master ve takım müşteri değerlerini sağlama adına Koşu Gereksinim Listesi'nde değişiklikler yapabilir.

4.5.3 Koşu Azalan Zaman Grafiği

Koşu Planlama Toplantısında takım koşu için gerçekleştirilecek gereksinimleri belirler, görev tanımlarını çıkarır ve koşunun başarılı olması için gerekli süre tahminlerini yapar. Tüm Koşu Gereksinim Listesi elemanlarının tahmini eforları toplam gereksinim eforudur. Koşu ilerleyişinde görevler tamamlandıkça Scrum Master kalan zamanı yeniden hesaplar ve toplam gereksinim eforunda devamlı bir azalma görünür. Eğer koşunun sonunda toplam koşu gereksinim eforu sıfır olduysa, koşu başarılı olmuştur [35].

Koşu Azalan Zaman Grafiği (Örnek Şekil 3.5), takıma ilerleyişini göstermek ve koşu sonunda taahhüt ettiği işleri gerçekleştirip gerçekleştiremeyeceğini göstermeye yardımcı olması için tasarlanmıştır [33].

Klasik biçimi koşudan önce takım üyeleri görevler için süre tahminlerini yaparlar ve günlük efor girişlerini yaparak, kalan efor süreleri hesaplanır ve bu şekilde grafik oluşmuş olur [33].



Şekil 3. 5 Koşu Azalan Zaman Grafiği örneği [35]

4.5.4 Ürün ve Dağıtım Azalan Zaman Grafiği

Proje gidişatında kullanılan tüm zaman döngüleri azalan zaman grafiği ile incelenebilir.

Ürün Azalan Zaman Grafiği tüm gereksinimlerinin süre tahminleri ile yapılan işler üzerinden hesaplanır.

Dağıtım Azalan Zaman Grafiği zamana bağlı olarak toplam ürün gereksinim tahminini verir. Tahmin edilen efor içindeki takımın iş listesi ve organizasyonun kararlaştırdığından oluşur. Zaman üniteleri genellikle koşulardır. Ürün Gereksinim Listesi tahminleri, dağıtım planlamasında ve yaratıldıkları zaman yapılır. Ürün Gereksinim Listesi düzenlemesinde gözden geçirme ve revizyon yapılır. Bununla birlikte istenildiği zaman bunlar güncellenebilir [28].

Yön çizgisi kalan işin değişikliği baz alınarak çizilir [28].

4.6 Scrum'da Yanlış Yapılanlar

Scrum'da bazen işler ters gidebilir. Bazı küçük sorunları büyümeden çözmek gerekir.

Aşağıda Scrum'da yapılan genellikle yapılan yanlışlıklar ve çözüm önerileri sıralanmıştır.

Ritm Kaybı

Belirti: Koşular hep aynı uzunlukta değil.

Tartışma: İyi ilerleyen bir Scrum projesinde takım doğal bir ritm oluşturur. Her Koşu aynı uzunlukta olmalıdır. Her Koşu gereksinimlerin tartışıldığı ve koşunun planlandığı Koşu Planlama Toplantısı ile başlar. Her koşu takımın potansiyel ürün artışı yaptığı bir Koşu Gözden Geçirme ile sona erer. Proje ritmi Günlük Scrum Toplantısı ile sağlanır [37].

Eğer koşu bazen 2 hafta bazen 4 hafta sürerse doğal ritm asla sağlanamaz. Koşu süresi takımlar tarafında değiştirilebilir seçildiği zaman Koşu Gereksinim Listesi için doğru süreyi seçmek zordur, bazen koşuda taahhüt edilen tüm elemanlar tamamlanamayabilir [37].

Konuşan Takım Dışı Bireyler

Belirti:Takım dışındaki bireylere Günlük Scrum Toplantıları'nda soru sorma ve gözlem yapma izni verilir.

Tartışma:Günlük Scrum toplantısında sadece takım üyelerinin konuşmasına izin verilir. Diğer kişiler gözlem yapabilir ama konuşamazlar. Diğer insanları konuşmasına izin vermek kaygan bir eğim gibidir. Bazen yararlı şeyler söyleyebilirler ama bu duruma izin verilirse daha sonra sürece zarar verebilecek söylemlere engel olunamayabilir [37].

Kayıp Takım Üyeleri

Belirti: Tüm Takım üyeleri günlük Scrum toplantısına katılmaz [37].

Tartışma: Günlük Scrum Toplantıları'nın aynı zamanda aynı yerde olması proje ritmi sağlanması ve artırılması açısından önem taşımaktadır. Eğer çok fazla üye sıkça Günlük Scrum Toplantısı'na katılmıyorsa toplantı daha az yararlı olacak ve 3 soruyu cevaplamada sapmalar yaşanacaktır [37].

İyi bir Günlük Scrum Toplantısı'nda toplam süre 15 dakikayı aşmamalıdır çünkü zaman tüm takım üyeleri için değerlidir. Bazı zamanlar pratikte yaklaşan dağıtım tarihleri nedeniyle takım üyeleri toplantılara katılmamazlık yapabilirler. Ama bu durum kronik bir hal alırsa, süreç için olumsuz bir etken olur [37].

Kalıcı İmzalar

Belirti: Takımın bir koşusunun Azalan Zaman Grafiği'nde görülen ani dalgalanmalar diğer koşularda da görülebilir.

Tartışma: Agile Software Development with Scrum'da Ken Schwaber ve Mike Beedle her takımın kendi özelliğinin olduğunu belirttiler. Bazı takımlar koşulara çok fazla iş getirirler bazıları çok az getirir. Ancak asıl önemli olan takımların geçmiş koşulardan geri dönüş alarak geri besleme ile öğrenim gerçekleştirilmesidir [37].

Scrum Master iş ataması yapar

Belirti: İş Scrum Master tarafından atanır, takım tarafından kabul edilir.

Tartışma: Kendi kendine organize olma Scrum'ın temel prensiplerinden biridir. Eğer Scrum Master görevleri takıma dağıtıcak olursa bu takımın sorumluluk alma yeteneğini baltalar ama eğer takımlara bu konuda insiyatif ve güven verilirse hedefe varma açısından daha sorumlu hissetmeleri sağlanır. Takım kendi işinde tüm kontrolün tamamen kendinde olduğunu hissetmelidir [37].

Günlük Scrum Toplantısı Scrum Master içindir

Belirti: Günlük Scrum Toplantısı sanki takım üyelerinin Scrum Master'a yaptığı bir durum güncellemesi gibidir [37].

Tartışma: Bazen günlük Scrum Toplantısının sadece Scrum Master için olduğu düşünülür [37].

Günlük Scrum toplantısının iki temel amacı vardır.

- Projedeki herkesin arasında bir koordinasyon mekanizmasının sağlanması [37]
- Tüm takım üyelerinin verdikleri taahhütler ile ilgili ne durumda olduğunu göstermesidir [37]

Uzman İş Rolleri

Belirti: Proje takımı Mimar, Tasarımcı, Veritabanı Yöneticisi, Testçi gibi uzman rollere ayrılmıştır [37].

Tartışma: Takım, bu işte biz hep birlikteyiz fikrine sahip olmaldır. Bu bazı zamanlar takımı özel yeteneklere sahip kişiler varsa zayıflatır [37].

Herkesin günümüz karmaşık teknolojilerinde bir konuda uzmanlaşmaması beklemek zordur. Ancak takım da kesin rollere ayrılmaksızın kişilerin ilgili olduğu konulara eğilmesi beklenir [37].

BÖLÜM 4

SCRUM SÜRECİ İŞLETİM VE YÖNETİM UYGULAMASI: SCRUMMAPP

Yazılım proje yönetiminde tercih edilen süreç modelleri kadar projelerin ve modellerin nasıl yönetildiği ve işletildiği de önem arz eder.

Organizasyonlar mevcut süreçlerinden yeni bir sürece geçiş yaptıkları zaman genelde sıkıntılarla karşılaşırlar. Bu sıkıntılar eğer geçiş yapılan süreç bir çevik süreç ise daha büyük boyutta yaşanır.

Bunun temel sebepleri;

- Çevik mantığa geçiş kolay değildir. Bugüne kadar gelinen süreçte geleneksel yazılım geliştirme süreçlerinde görev almış insanların kafa yapısı olarak klasik yöntemlere daha yatkın olduğu aşikardır. Bu düşüncedeki bireylerin çevik mantığa yaklaşımları ilk başta olumlu olmaz. Eski yöntemlerde, geliştirme harici eforları çok harcayan, günlerce hazırlık ve dökümantasyon ile uğraşan bireyler yeni sürece uyum sorunu yaşarlar. En büyük sıkıntı önceden kendilerini müşteriden izole ederek geliştirme yapan bireylerin sürecin her safhasında müşterinin dahil olduğunu görerek, müşteri ile beraber çalışma zorluğu yaşamasıdır.
- Yeni bir süreç eğitim gerektirir. Her yeni süreç geçişi takımlar için büyük risk teşkil eder. Bu nedenle sürece hızlı adaptasyon için etkili bir eğitim ve başarılı süreç yönetiminin gerçekleştirilmesi gereklidir.

- Yetkin uzman sayısı yeterli değildir. Süreçlerde iş kuralları, süreç kuralları gibi konularda yetkin kişi azlığı süreçlerin bambaşka bir hal almasına ve etkin kullanılmamasına neden olmaktadır.

ScrumMApp uygulaması öncelikli olarak sürece yeni geçiş yapan organizasyonları hedef kitle olarak kabul etmiştir. Mevcut yetkinlik eksikliklerini süreç ile bilgiler vererek gidermeyi ve kolay önyüzü, doğru iş kuralları ile süreci doğru yönettirmeyi hedefler.

5.1 Mevcut Durum Analizi

Scrum, Türk yazılım şirketlerinin son zamanlarda tercih etmeye başladığı bir yazılım geliştirme süreç modeli olmuştur. Ancak şirketler genelde süreç geçişinde ve uygulamada bazı sorunlar yaşıyorlar.

Yürütülen tez çalışmasında Türkiye'deki Scrum kullanan çeşitli sektördeki yazılım geliştirme şirketleri ve danışman şirketlerde çalışan bireyler ile yapılan anket ve yüzyüze görüşmeler ile organizasyonlardan şu durum bilgileri alınmıştır;

- Yeni bireyler, sürece adapte olmakta zorlanıyor.
- Yetkin olmayan Scrum Master'lar nedeniyle süreç doğru kuralları ile işletilemiyor.
- Scrum doğası ve iş kurallarını benimsemeyen süreç yönetim yazılımları, sürecin takibi ve yönetiminde başarılı değil.
- Sürece tam dahil olamayan ürün sahipleri, isteklerini tam olarak iletemiyor ve süreci takip etmiyorlar.
- Süreç çoğu organizasyonda Excel ve yapışkan kağıtlar ile takip ediliyor, gerçek zamanlı sonuçlar alınamıyor.

5.2 Gereksinimlerin Alınması

Gereksinimler yüzyüze yapılan görüşmeler ve internet üzerinde yapılan anketler vasıtasıyla alınmaya çalışılmıştır. Aşağıdaki sorular ile organizasyonların Scrum sürecine bakış açısı, sürecin iyi, kötü ve eksik yönlerinin belirlenmesi, bir aracın sürece ne katacağını ve bir süreç yönetim aracından beklenen özelliklerin öğrenilmesi hedeflenmiştir.

Bu anket yurtiçinde Scrum'da eski veya yeni farketmeksizin hatta Scrum danışmanlığı yapan şirket üyeleri olmak üzere Bankacılık, Telekomünikasyon, Yazılım vb. sektörlerde çalışan 30 katılımcı ile gerçekleştirilmiştir.

Anket soruları Çizelge 4.1'deki gibidir;

Çizelge 4. 1 Anket soruları

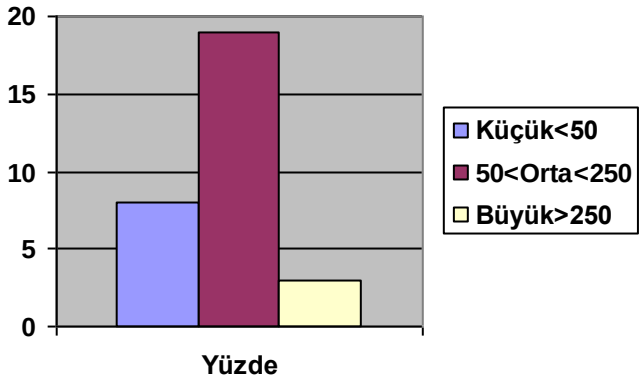
Çalıştığınız şirketin adı?	
Çalıştığınız şirketin sektörü?	
Organizasyonunuzun büyüklüğü?	Küçük<50 Orta 50<x<250 Büyük >250
Scrum'dan önce kullandığınız yazılım süreçleri hangileridir?	
Şirketinizdeki Scrum takımı sayısı?	1-5 5-10 10-20 20-50 50+
Scrum'ı seçmenizdeki ana etken nedir?	Üretim hızını artırması Değişikliğe yanıt verme yeteneği kazandırması Kaliteyi artırması Projeleri zamanında planlamayı ve bitirmeyi sağlaması Diğer
Scrum'da sizi en çok zorlayan öğe nedir?	Yönetim desteği Takım motivasyonu Doğru Ürün Sahibi'ni bulmak Test kalitesini sağlamak Çapraz fonksiyonlu takımları bulmak Diğer
Çevik süreçlerle en uyumlu olduğunu düşündüğünüz öğe nedir?	Ürün sahibi Değişiklik yönetimi Şirket yönetimi Yazılım geliştirme yaşam döngüsü Diğer
Kalite yönetimini en çok tetikleyen öğe nedir?	Müşteri Takım Scrum Master Şirket yönetimi Test birimi Diğer

Takımlarda hangi yetkinliklere sahipsiniz?	Geliştirici İş analisti Testçi Ürün Sahibi Scrum Master
Ürün sahiniz bir iş analisti mi?	Evet / Hayır
Kullandığınız Ürün Gereksinim Listesi'nin özellikleri nelerdir?	Kullanıcı hikayelerinden oluşur Önceliklendirilmiştir. Risk içerir İş değeri içerir Her koşu sonrası değişir (Genellikle) Herkes katkıda bulunabilir Ürün Sahibi katkıda bulunabilir Diğer
Scrum sürecinin yönetiminde kullandığınız özel bir araç var ise belirtiniz?	
Scrum sürecine uygun özel bir aracın süreç yönetimini,ürün kalitesini ve üretim hızını artıracağını düşünüyor musunuz?	Evet / Hayır
Bir Scrum proje yönetim aracı geliştiriyor olsaydınız nelere dikkat ederdiniz?	Kolay kullanılabilirlik Esneklik Takip edilebilirlik Öğretici bilgiler Diğer
Scrum'dan bir özelliğini çıkarmak, uygulamamak isteseydiniz bu hangisi veya hangileri olurdu?	Koşular Koşu Planlama Toplantısı Günlük Scrum Toplantısı Koşu Gözden Geçirme Toplantısı Koşu Kapatma Toplantısı Gereksinim güncelleme Dağıtımlar Diğer
Scrum'a bir özellik eklemek isteseydiniz bu ne olurdu?	
Scrum Master'ın süreçteki en önemli görevi sizce nedir?	
Ürün Sahibi'nin süreçteki en önemli görevi sizce nedir?	

5.3 Anket Sonuçları ve Değerlendirmeler

Organizasyon büyüklüğü sorusu anket katılımcılarının organizasyonel büyüklük olarak ölçme amacı ile yapılmıştır. Ayrıca organizasyon büyüklüğü süreç ilişkisini de kurma amaçlı bu bilgi alınmıştır.

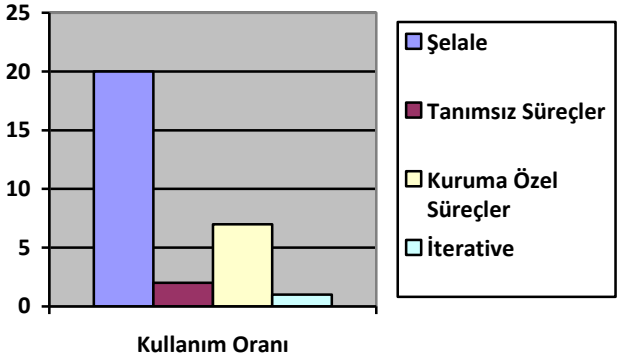
Şekil 4-1'de görüldüğü üzere anket katılımcılarımızın çoğunu orta ölçekli (50 ile 250 çalışanı) şirketlerde çalışan bireyler oluşturmaktadır. Bu küçük işletmeler çalışan bireyler yerine daha kurumsal yapıya ulaşmış kalabalık şirketlerdeki katılımcılara ulaşmış olduğumuz anlamına gelir ki bu çalışma başarısı adına iyi bir göstergedir.



Şekil 4. 1 Organizasyon büyüklüğü

Anket katılımcılarından Scrum'dan önceki kullandığı süreçler alınarak genel anlamıyla Scrum öncesi süreçleri ve hangi süreçleri kullanan bireylerin Scrum'a geçmeye daha yatkın olduğu anlaşılmaya çalışılmıştır.

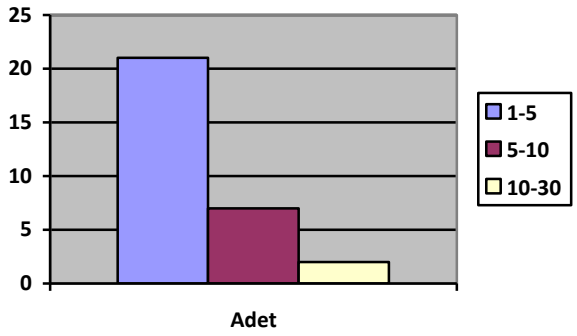
Şekil 4.2'deki sonuçlar gösteriyor ki katılımcıların büyük çoğunluğu Scrum'dan önce Şelale gibi geleneksel süreçler ile yazılım geliştiriyordu. Bazı organizasyonların kendi süreçlerini oluşturduğu bazılarının ise serbest geliştirme yaptığı görülüyor. Geleneksel süreçlerdeki organizasyonların artık yeni bir arayış içerisinde olduğunu bu sonuçlardan görmekteyiz.



Şekil 4. 2 Kurumda önceden kullanılan süreçler

Kurumdaki Scrum takımı sayısı ile katılımcılardan organizasyonlardaki Scrum'ın kullanım şekli tespit edilmeye çalışılmıştır. Scrum kurumda deneme amaçlı birkaç ekiple mi yürütülüyor yoksa tam organizasyon çevikleştirilmeye çalışıyor bu anlaşılmaya çalışılmıştır.

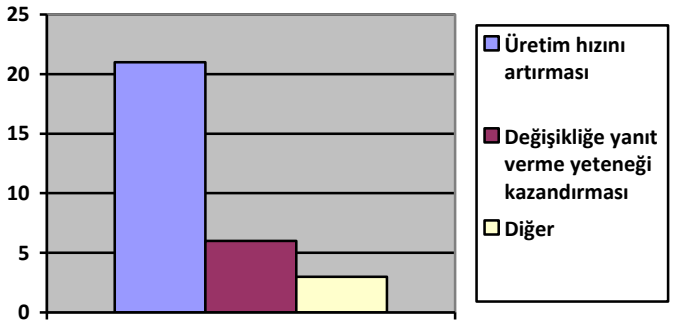
Aldığım sonuçlar (Şekil 4.3) daha şirketlerin bu hususta yolun başında olduğunu gösteriyor, katılımcıların büyük bir kısmı organizasyonlarında az sayıda Scrum takımı olduğunu belirtti. Bu da bizim hedef kitlemiz olan Scrum sürecine yeni başlayan organizasyonların fazlalıkta olduğunu görmemizi sağladı.



Şekil 4. 3 Kurumdaki scrum takımı sayısı

Organizasyonların Scrum'a geçiş yapma sebeplerini alarak Scrum'ı seçmedeki temel etkeni almaya, akademik çalışmalar ile gerçek yazılım geliştirme ortamının aynı paralelde olup olmadığını görmeye çalıştım.

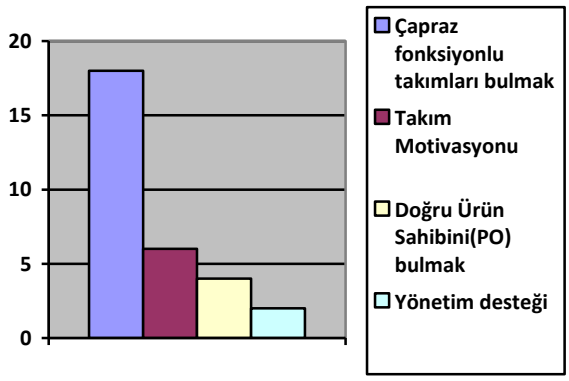
Şekil 4.4'de görülen sonuçlar gösterdi ki Scrum'ın temel hedefi ve taahhütü olan üretim hızını artırması ve değişikliklere yanıt verme yeteneği kazandırması organizasyonlarında bu süreci seçme sebebi olmuş.



Şekil 4. 4 Organizasyonların Scrum'ı seçme sebepleri

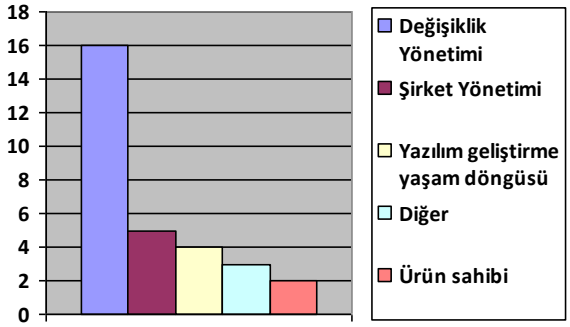
Scrum'da katılımcıları en çok zorlayan öğe aslında bu süreç yönetiminde öncelikli olarak çözüm getirilmesi gereken bir sorundur.

Şekil 4.5'deki sonuçlar bize gösterdi ki takımlar yetkin bireyler bulmada ve takım motivasyonu sağlamada sorunlar yaşıyorlar. Süreç işletim sisteminde süreç eğitimi gerçekleştirerek bir nebze yetkin birey sorunu çözmede organizasyonlara yardımcı olabilir, doğru işletilen ve başarılı yürüyen bir proje ise takım motivasyonunun kendiliğinden kazanılmasına yardımcı olabilir. Bu şekilde temel sorunlara çözüm getirmeyi hedefledik.



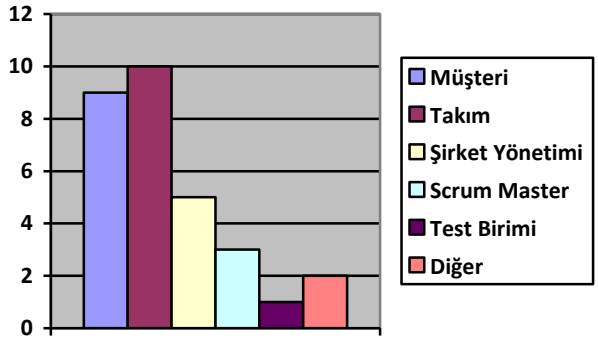
Şekil 4. 5 Scrum'da sizi en çok zorlayan öğeler

"Scrum'ı uygularken en uyumlu bulduğunuz öğe nedir?" sorusuna Şekil 4.6'da da görüldüğü gibi değişiklik yönetimi cevabını almak aslında beklediğim bir sonuç oldu. Scrum temel olarak bunu hedefler.



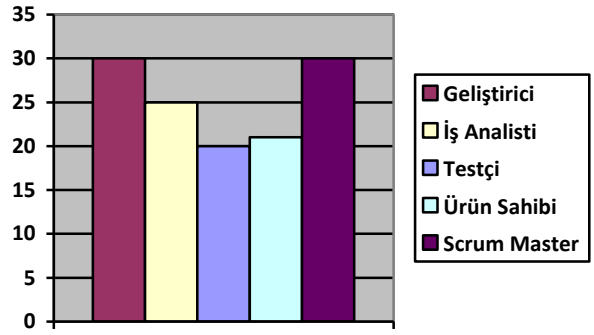
Şekil 4. 6 Çevik süreçlerle en uyumlu olduğunu düşündüğünüz öğeler

Scrum'da kalite anlamında hangi öğelerin etkili olduğu sorusuna aldığım cevaplar gösterdi ki (Şekil 4.7) müşteri ve takım ürünün kaliteli olması anlamında diğer öğelerden daha önde ve etkili konumda.



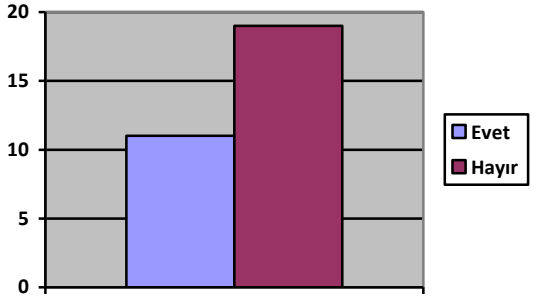
Şekil 4. 7 Kalite yönetimini en çok tetikleyen öğeler

Organizasyonlarda ki takımların becerilerini belirleyebilme adına sorduğum sahip olduğunuz yetkinlikler sorusunun sonuçları (Şekil 4.8) gösterdi ki takımlar genelde tam kadro tüm rolleri barındıran bir aile olarak Scrum süreci içerisinde bulunuyor. Bu zaten projenin başarılı olması için temel gerekliliklerden birisidir.



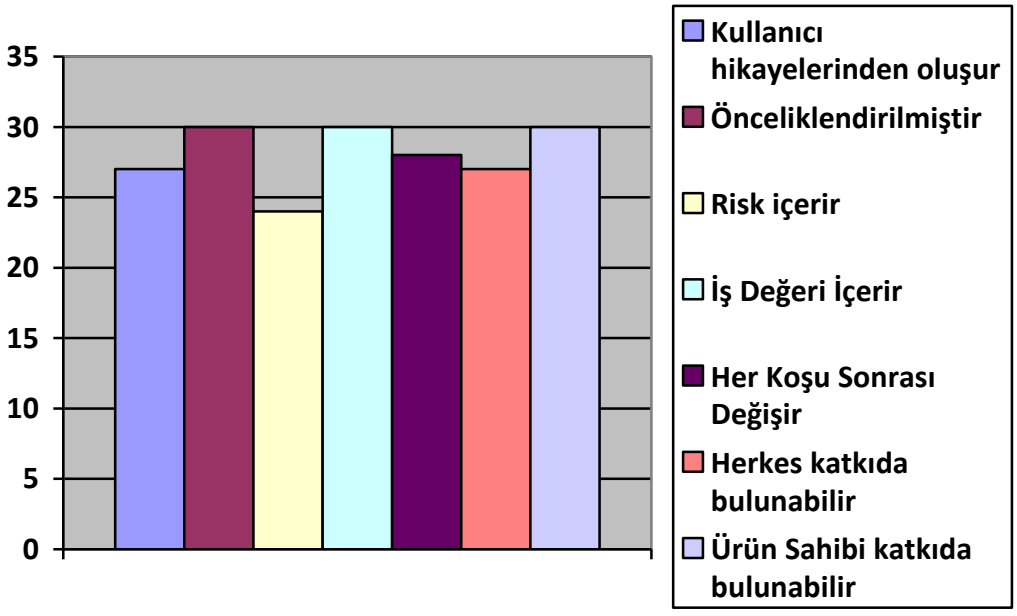
Şekil 4. 8 Takımlarda sahip olunan yetkinlikler

Organizasyonların Scrum'da ki Ürün Sahibi rolüne bakış açısının belirlenmeye çalıştığı soruya alınan cevaplara (Şekil 4.9) göre organizasyonlar yavaş yavaş çevik olma yolunda iyi yönde değişiklikler gösteriyorlar. Kapsamlı analiz yapan bireyler yerine ürünü sahiplenen, beklediği ürünün ne olduğu bilen kişiler Ürün Sahibi olarak görev alıyor.



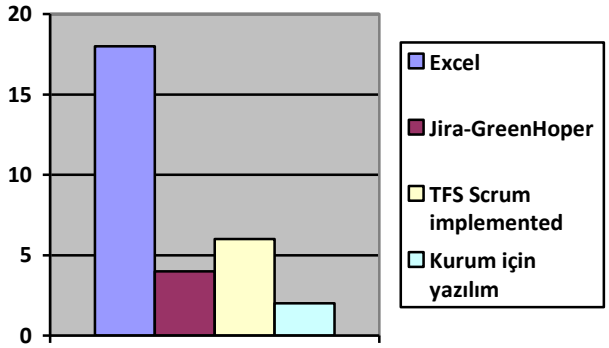
Şekil 4. 9 Ürün Sahibi bir iş analisti mi?

Kullanıcılara "Ürün Gereksinim Listesi'nin özellikleri nelerdir?" sorusuna cevap seçeneği olarak aslında bir listenin sahip olması gereken tüm özelliklerini sunarak organizasyonların kullanmadığı bir özellik varsa onu belirlemeye çalıştık. Aldığımız cevaplar (Şekil 4.10) bu listenin genelde doğru özellikler taşıdığını gösterdi.



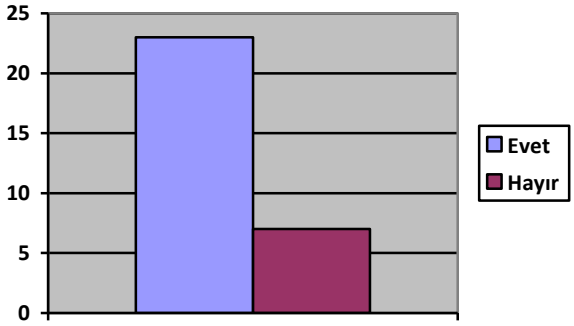
Şekil 4. 10 Ürün Gereksinim Listesi'nin özellikleri

Anketimizde bulunan en can alıcı soruya aldığım cevaplar (Şekil 4.11) gösteriyor ki organizasyonlar Scrum süreçlerini büyük oranda sürece özel geliştirilmeyen yardımcı araçlar ile yönetmeye çalışıyor. Bu da gerçekleştireceğim ürünü katabileceği katkılarla organizasyonları Scrum ile başarıya götürmeye aday bir ürün haline getiriyor.



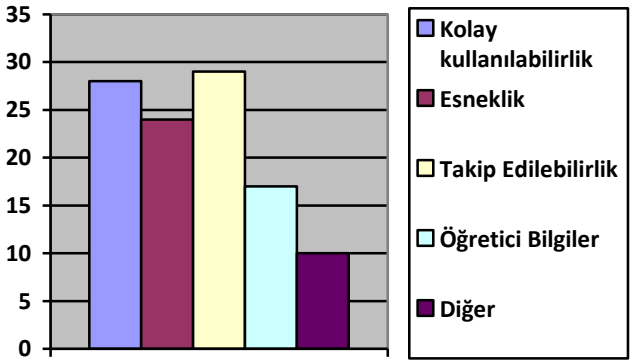
Şekil 4. 11 Scrum'ı yönetmek için kullanılan özel araçlar

Hedefimiz olan iş ile ilgili katılımcılara yönelttiğimiz soruya aldığımız cevaplar gösteriyor ki (Şekil 4.12) katılımcılar özel bir süreç yönetiminin başarı getireceği fikrini benimle paylaşıyorlar.



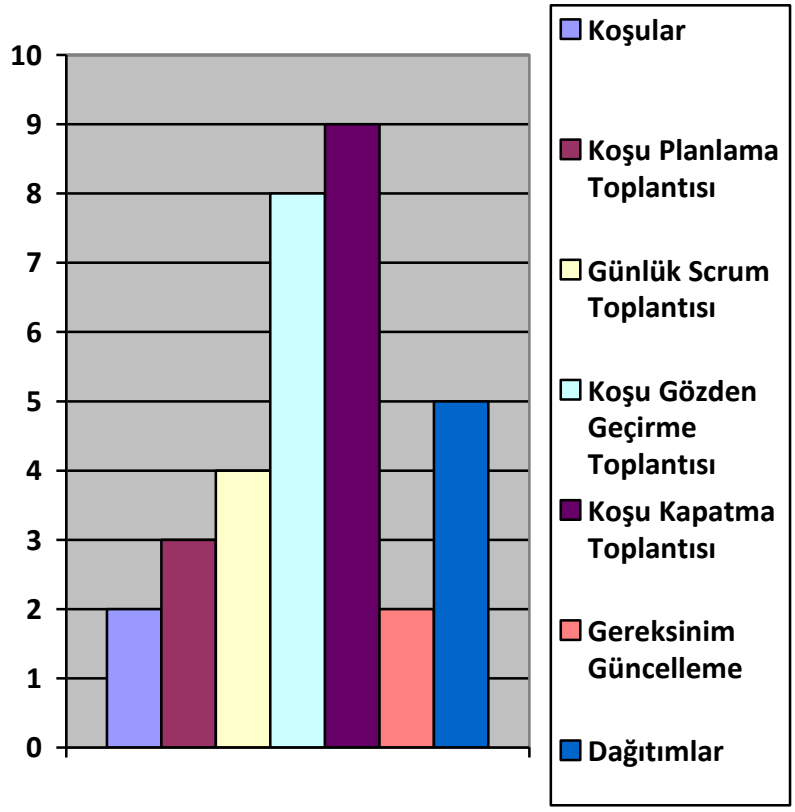
Şekil 4. 12 Scrum sürecine özel aracın katkısı olur mu?

Geliştirilecek özel araçlardan beklentilerini sorduğum zaman aldığım cevaplardan (Şekil 4.13) kolay kullanılabilirlik, esneklik ve takip edilebilirliğin talep edilen temel özelliklerden olduğunu gösterdi.



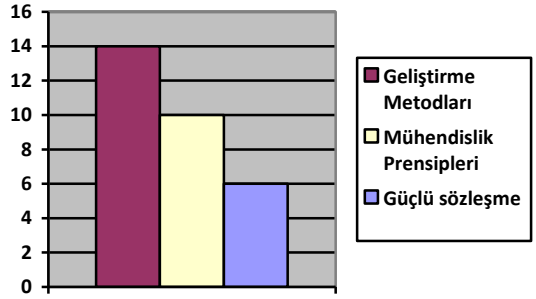
Şekil 4. 13 Scrum aracı geliştiriyor olsaydınız nelere dikkat ederdiniz?

Scrum ile organizasyonların düşüncelerini aldığımız kısımlara geçiş yaptığımız soruların birincisi “Beğenmediğiniz Scrum öğesi nedir?” oldu. İlginç cevaplar aldığımız sonuçlarda (Şekil 4.14) kişiler genelde Koşu Planlama Toplantısı ve Koşu Gözden Geçirme toplantısını gereksiz bulmaktadır.



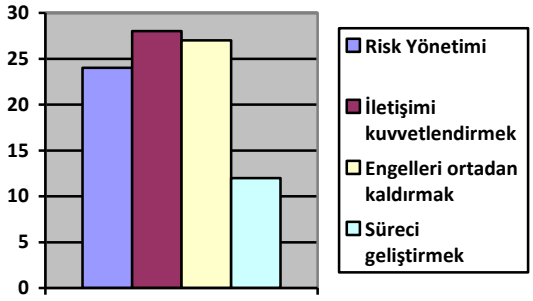
Şekil 4. 14 Scrum'dan bir özelliğini çıkarmak,uygulamamak isteseydiniz?

Kendi Scrum’ınızı yaratın sloganı ile sorduğumuz soruya aldığımız cevaplar (Şekil 4.15) gösterdi ki kişiler Scrum’dan en çok geliştirme ile ilgili de katkılar bekliyor. Sadece planlamaya yönelik çalışan Scrum bu konuda başka bir çalışma ile geliştirilebilir.



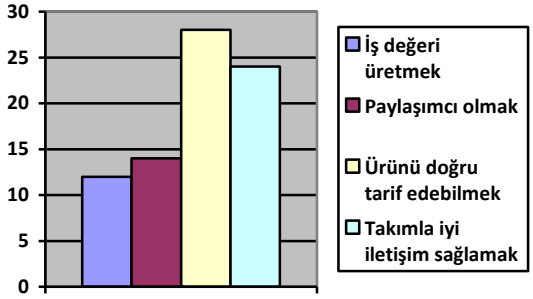
Şekil 4. 15 Scrum'a bir özellik eklemek isteseydiniz bu ne olurdu?

Kişi ve organizasyonların Scrum Master rolüne bakış açısını almaya çalıştığımız sorunun cevapları (Şekil 4.16) kişilerin Scrum Master'den temel görevlerini yerine getirmesini beklediğini gösteriyor.



Şekil 4. 16 Scrum Master'ın süreçteki en önemli görevi sizce nedir?

Kişi ve organizasyonların Ürün Sahibi rolüne bakış açısını almaya çalıştığımız sorunun cevapları (Şekil 4.17) kişilerin Ürün Sahibi'nden temel olarak ürünü doğru tanımlamasını ve takımla birlikte çalışabilmesini beklediğini gösteriyor.



Şekil 4. 17 Ürün Sahibi'nin süreçteki en önemli görevi sizce nedir?

5.4 Analiz

Süreç geçişindeki olumsuz durumlar göz önüne alındığı zaman, gerçekleştirilen anket değerlendirmeleri ve yapılan yüzyüze görüşmeler sonrası organizasyonların sıkıntılarını en aza indirecek, sürece yeni organizasyonlara hızlı uyum imkanı, eskilere de kendi süreçlerini iyileştirme imkanı sağlayacak bir yazılım sistemi geliştirme fikri doğmuş ve tez çalışmasının temelini oluşturmuştur.

Geliştirdiğim yazılım sistemi ile Scrum sürecini kullanan organizasyonların tüm Scrum araç ve zamanlamaların yönetimi, takibi ile birlikte, anlık raporlar sağlanıp, proje ilerleyişinin, toplantıların ve kişilerin takibi sağlanacaktır. Ürün Sahibi'nin projenin daha fazla içinde ve bilgi sahibi olması sağlanarak, istediği gereksinimlere sahip sonuç ürün çıkması ihtimali artırılacaktır.

Mevcut iş analizi, Scrum ile yapılan akademik araştırmalar ve organizasyonların tecrübeleri ile iş kuralları seti ve süreç modeli çıkarılmıştır.

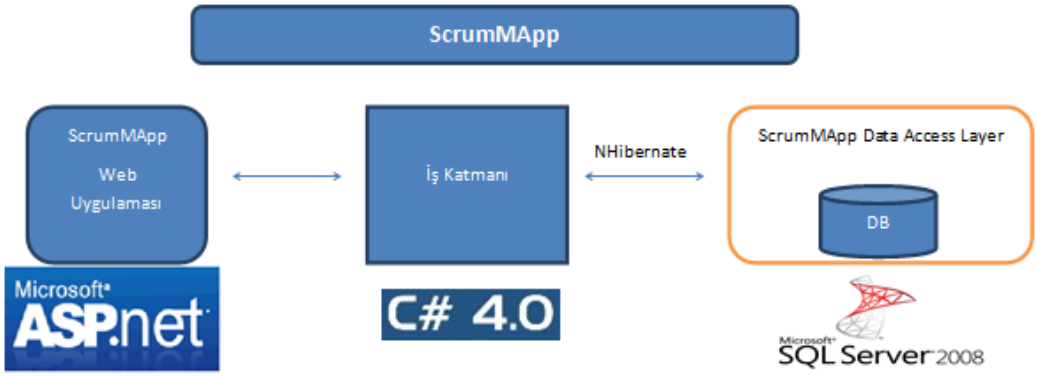
Scrum sürecini yönetecek bir yazılım sisteminden beklenen temel gereksinimler şunlardır;

- Tüm süreç paydaşlarını yazılım sistemine kabul etmelidir. (Müşteri, Ürün Sahibi, Scrum Master, takım, diğer tüm proje paydaşları)
- Gereksinimlerin oluşturulup, önceliklendirilmesini, değiştirilmesini sağlaması ve efor tahminlerini alıp, saklaması gerekmektedir.
- Tüm zaman döngülerinin oluşturulması, yönetilmesi ve takibini sağlamalıdır.(Proje, dağıtım, koşu, görev, olay)
- Online olarak tüm zaman döngülerinin Azalan Zaman Grafiği gözlenebilmelidir.
- Tüm süreç paydaşları görevlere efor girişi yaparak, anlık süreç ilerleyiş bilgisinin alınması gerekmektedir.
- Dosya yönetimi ile proje ilgili döküman, kod vb. dökümanların sistem içinde saklanması gerekmektedir.
- Süreç için önemli bir yeri olan toplantıların yönetimi ve kontrolü sistemden sağlanmalıdır.
- Sürecin doğası gereği mevcut yazılım proje yönetimi yazılımlarından farklı olarak kişilere görev atama yerine görev seçme imkanı sağlamalıdır.
- Süreç kurallarının doğru işletilmesini sağlamalıdır. (Örnek:"Koşu başlatılabilmesi için Koşu Planlama Toplantısı yapılmalıdır" kuralı gibi)
- Süreçle ilgili bilgiler vererek bir yandan süreç eğitimi de sağlaması gerekmektedir. (Tanımlar, açıklamalar)

5.5 Tasarım

Mevcut yazılım analizi, iş kurallarının belirlenmesinden sonra geliştirilecek yazılım sisteminin mimarisi belirlenip, basit ve detaylı tasarımı gerçekleştirilmiş, kullanıcılardan alınan beslemeler neticesinde çevik anlayış gereği çok fazla efor sarfedilmeden yazılım geliştirilmeye başlanmıştır.

Uygulamanın yazılım mimarisi Şekil 4.18'deki gibidir;



Şekil 4. 18 ScrumMApp yazılım mimarisi

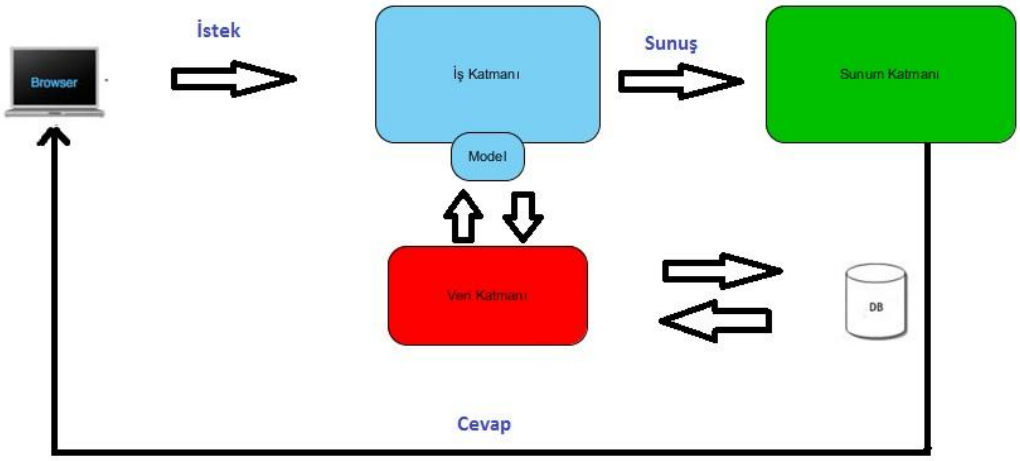
ScrumMApp yazılım sistemi dağıtık takımlar veya uzak Ürün Sahibi gibi faktörler düşünülerek web uygulaması olarak geliştirilmiştir. Web uygulaması olarak geliştirme seçiminin temel sebebi bu tür uygulamaların her yerden kurulum maliyetsiz kullanılabilmesidir.

Asp.Net uygulaması olarak C# dilinde geliştirilen uygulamada veri tabanı olarak Sql Server 2008 kullanılmıştır.

Uygulama yerel ağ veya internet üzerinde herhangi bir IIS, .Net framework ve Sql Server yüklü bilgisayarda çalışabilecek şekilde ayarlanmıştır. İstemcilerin ise sadece ağ bağlantısı ve tarayıcıya ihtiyacı vardır.

Uygulama nesne tabanlı(object oriented) ve Şekil 4.19'da görüldüğü gibi 3 katmanlı mimaride geliştirilmiştir.

3 katmanlı mimari nesneye yönelik programlama ile başlayan veri tabanı programlarının temelini oluşturan bir akımdır.



Şekil 4. 19 ScrumMApp 3 katmanlı mimari

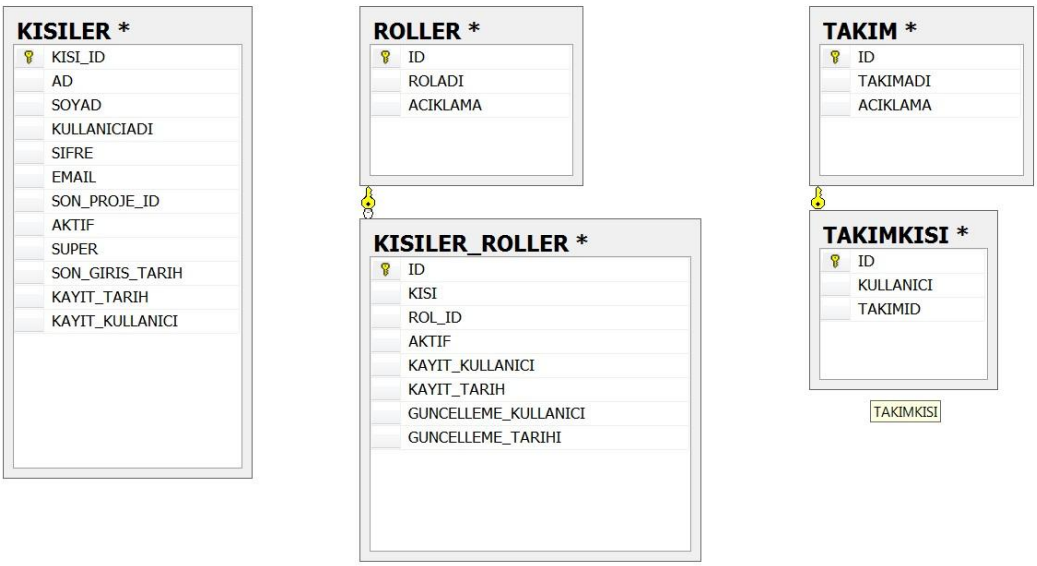
Veri Katmanı

Yazılan uygulamaların çoğunun veritabanına bağımlı çalıştığını düşünürsek uygulamalarda öncelikle veriye ulaşım katmanına ihtiyaç duyulur. Verinin veritabanı sisteminden getirilmesi ve veritabanı sistemine gelen verilerin eklenmesi için oluşturduğumuz ilk katmanımız veri katmanı olur.

Veri katmanı için öncelikle uygulama gereksiniminin yapılıp, senaryolar ile kişilerin hangi işlemleri gerçekleştireceği, hangi nesnelerin kullanılacağı ve bu nesnelerin hangi özellikleri barındıracağının analizi ile başlanmıştır.

Analiz sonucu belirlenen nesne sınıflarının tablo diyagramları aşağıdaki gibidir; (Bu diyagramlar aynı zaman da nesne modellerini de tanımlar)

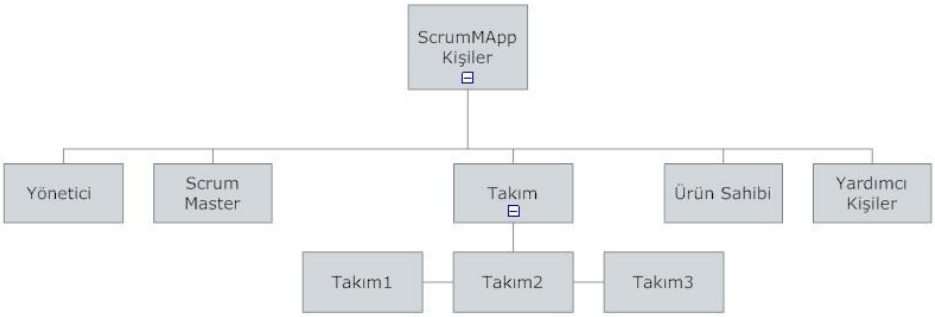
Kişi ve kullanıcı yönetimleri ile ilgili diyagramlar Şekil 4.20'deki gibidir.



Şekil 4. 20 ScrumMApp kişi yönetimi ile ilgili tablo diyagramları

KISILER: Kişi bilgilerinin tutulduğu tablodur. Uygulamaya giriş yapılacak kullanıcılar uygulama üzerinden bu tabloya kaydedilirler.

ROLLER: Uygulamadaki rol tanımlarının tutulduğu tablodur. Kişilerin yetkileri bu rollere göre belirlenmektedir. Uygulamadaki roller Şekil 4.21'deki gibidir;



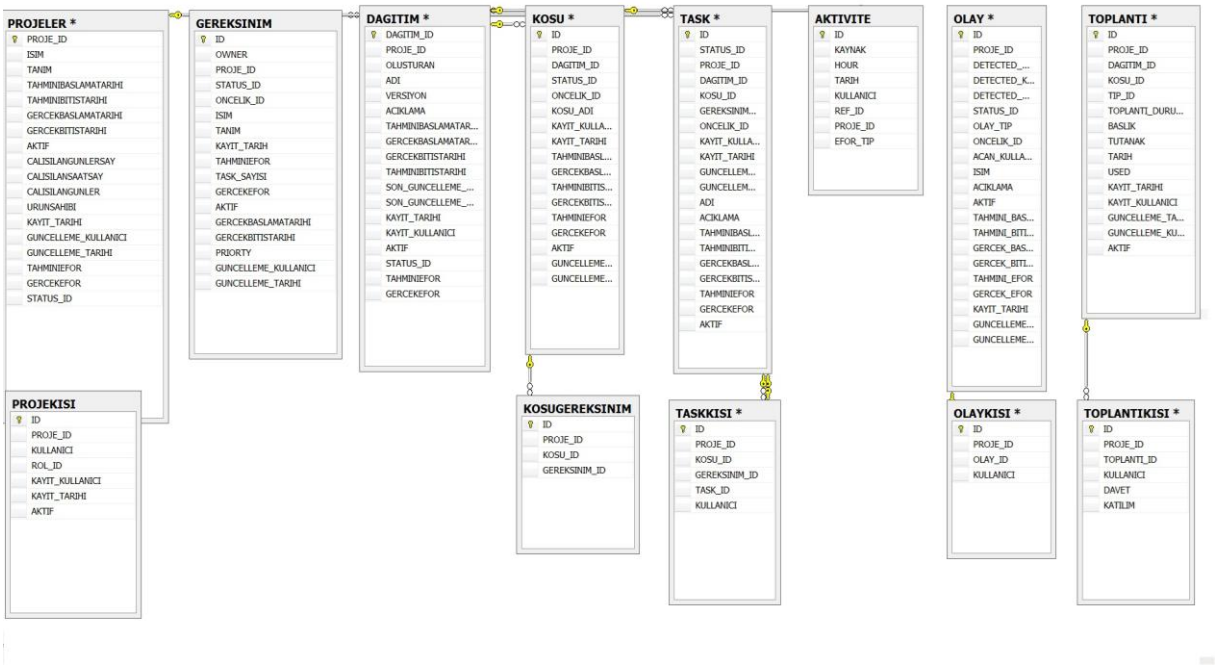
Şekil 4. 21 ScrumMApp rol tipleri

KISILER_ROLLER: Kişi rol ilişkisinin tutulduğu tablodur. 1-n ilişkisi vardır, kişi bir veya birden role tanımlı olabilir.

TAKIM: Scrum'da takım kavramı vardır. Süreç de bireyler bir takım altında veya bireysel tanımlanabilir. Takım kavramı bireylerin toplu proje atanmasında kolaylık sağlama ve takım takibi açısından kullanılmıştır.

TAKIMKISI: Takım kişi ilişkisini tutar. 1-1 ilişkisi vardır.

Süreç ile ilgili diyagramlar Şekil 4.22'de gösterildiği gibidir;



Şekil 4. 22 ScrumMApp süreç yönetimi ile ilgili tablo diyagramları

PROJELER: Projeler ile bilgilerin tutulduğu tablodur. Süreç yönetimi bu tanımlanan projeler üzerine olur.

PROJEKISI: Projede tanımlı kişilerin bilgisinin tutulduğu tablodur. 1-n ilişkisi vardır.

GEREKSINIM: Proje ile ilgili gereksinimlerin tutulduğu tablodur. Proje tablosu ile 1-n ilişkisi vardır.

DAGITIM: Proje de çıkılacak dağıtımların bilgilerinin tutulduğu tablodur. Proje tablosu ile 1-n ilişkisi vardır.

KOSU: Projede ki Scrum süreci için en önemli zaman döngüsü olan koşuların tutulduğu tablodur. Proje, dağıtım ile 1-n ilişkisi vardır.

KOSUGEREKSINIM: Koşu da gerçekleştirilmesi planlanan gereksinimlerin tutulduğu tablodur. Koşu ile 1-n ilişkisi vardır.

TASK: Koşu da yapılması taahhüt edilen gereksinimlerin artık takım üyesi tarafından işe dönüştürülmüş hali olan görevlerin tutulduğu tablodur. Proje, dağıtım, koşu ve gereksinim ile 1-n ilişkisi vardır.

TASKKISI: Görev de çalışacak kişi ilişkisinin tutulduğu tablodur. 1-n ilişkisi vardır. Bu tabloya Scrum Master veya takım üyeleri bireysel olarak görev seçerek ekleme yapabilirler.

AKTIVITE: Proje de gerçekleştirilen her türlü efor girişi (görev veya olay çözümü) bu tabloda saklanır. Efor raporları bu tablo üzerinden çekilir.

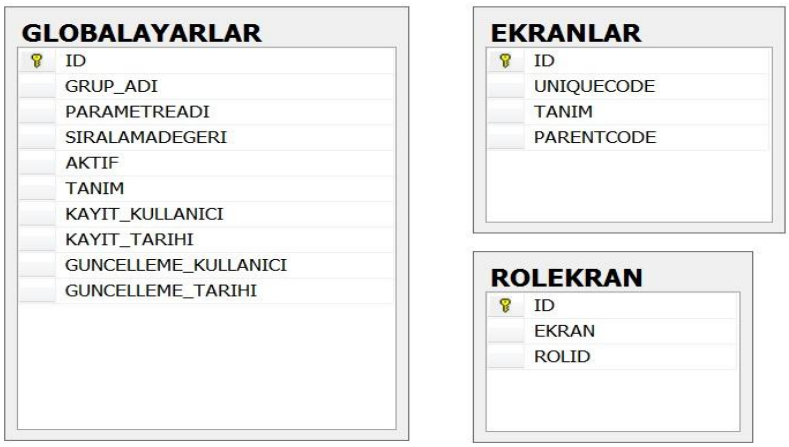
OLAY: Proje de ürün geliştirme esnasında karşılaşılan her türlü engel, hata, risk bu tablo üzerine kaydedilir.

OLAYKISI: Olay kişi ilişkisinin tutulduğu tablodur. 1-n ilişkisi vardır. Bu tabloya Scrum Master veya takım üyeleri bireysel olarak olay seçerek ekleme yapabilirler.

TOPLANTI: Projede ki toplantı bilgileri bu tabloda saklanır. Bu tabloya veri girişi sadece Scrum Master vasıtasıyla gerçekleştirilir.

TOPLANTIKISI: Toplantı katılım durumunun tutulduğu tablodur. Toplantı katılım raporları bu tablo yardımıyla çekilir.

Uygulamada ki genel ayalarların tutulduğu nesnelerin diyagramları aşağıdaki (Şekil 4.23) gibidir;



Şekil 4. 23 ScrumMApp sistem yönetimi ile ilgili tablo diyagramları

GLOBALAYARLAR: Uygulamanın daha esnek bir yapıda kurulması için sistem değişkenlerinin tutulduğu tablodur.

EKRANLAR: Uygulamadaki formların bilgilerinin tutulduğu tablodur.

ROLEKRAN: Uygulama da hangi ekranın hangi role açık olduğu bilgisinin tutulduğu tablodur.

Veritabanı ile iletişim nHibernate teknolojisi ile sağlanmıştır.

Geliştirilen her yazılımda yazılımcılar olarak veri depolama ihtiyacı duyarız. Günümüzde XML dosyalarından, ilişkisel veritabanlarına, meitn dosyalarına kadar veri depolamak için birçok alternatif mevcuttur. Bu birçok alternatife erişip veri üzerinde işlem yapmak için yine birçok farklı teknoloji yazılım profesyonelleri tarafından kullanılmaktadır.

Ado.Net vb. teknolojilerin aksine nHibernate bir O/RM çatısıdır. .Net ile yazdığımız sınıfların, ilişkisel veritabanı sistemleri ile eşleştirmesini (dönüştürülmesini) gerçekleştirir. Yani biz tablolarımıza karşılık gelen nesne ve haritalama dosyalarımızı oluşturup veritabanı sorgusu yazmadan bu teknoloji üzerinden veritabanı üzerinden veri işlemlerimizi gerçekleştirebiliyoruz.

İş Katmanı:

Veritabanından veriler veri katmanını aracılığı ile geldikten ya da veri katmanından veritabanına iletilecek veriler geldikten sonra bu verilerin düzenlendiği uygulamanın ihtiyaçlarına uygun değişime uğradığı katman ise iş katmanıdır.

Tablolarımızın nHibernate ile haberleşebilmesi için model sınıflarının oluşturulmasından sonra tüm gerçekleştirilecek işlemlerin hazırlanması bu katman içerisinde gerçekleştirilmiştir. İş katmanı veri katmanı ile haberleşerek sunum katmanına veriyi sunum katmanında istenildiği gibi hazırlandığı katmandır.

Sunum Katmanı:

Kullanıcının göreceği ve kullanıcının girdiği verilen alınacağı yada daha önceden girilmiş verilerin bir şekilde kullanıcıya gösterilmesi için gereken bir katmana daha gerek vardır. Kullanıcı ile iletişim halinde olan bir katman ve bu katman kullanıcının girdiği verileri

alıp bir alt katmana iletir ve veritabanından gelen verileri kullanıcıya gösterir. Bu katman da sunum katmanıdır.

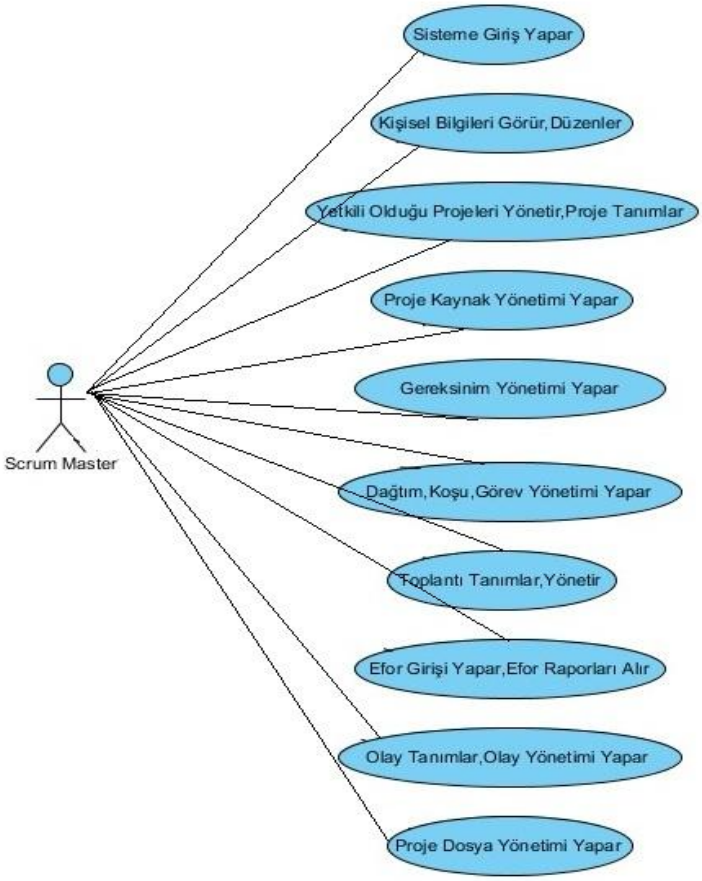
5.6 Uygulama

Bu çalışma süresinde geliştirilen uygulama ön analizler, tasarım, detaylı tasarımdan sonra gerçekleştirilen yazılım Scrum sürecini başarı ile yönetmeye aday bir uygulama olmuştur.

Sistem kullanıcısının bakış açısıyla, sistemin davranışlarının diyagramlarla modelleyerek rollerin sistemdeki işlerini senaryolar ile anlatarak açıklamaya çalışalım;

Scrum Master

Scrum Master'ın uygulamada gerçekleştirebileceği işlemler Şekil 4.24'deki diyagram ile gösterilmiştir.



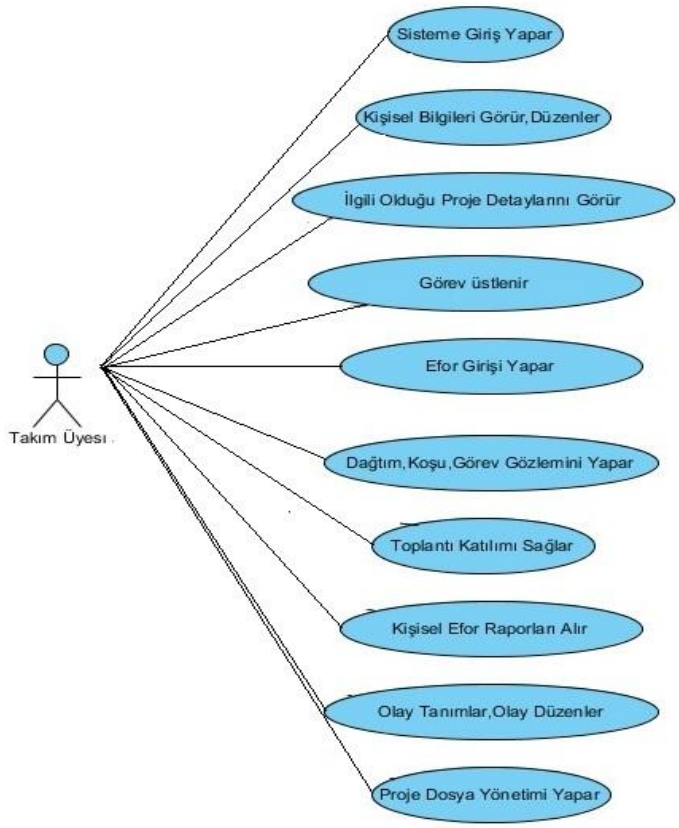
Şekil 4. 24 ScrumMApp'da Scrum Master

Scrum Master sürecin en önemli üyelerinden ve sürecin ilerleyişinden sorumlu birey olarak sistemde görevi gereği atandığı veya oluşturduğu projelerin takibinden ve sürecin doğru çalıştırılmasından sorumludur.

- Proje oluşturma veya görevli olduğu proje ile ilgili bilgileri düzenler.
- Proje kaynak yönetimini yapar, projede çalışacak bireyleri ve Ürün Sahibi'ni belirler, gerekli yetki atamalarını yapar.
- Ürün Sahibi ile beraber Ürün Gereksinim Listesi'ni düzenler.
- Takım üyeleri ile birlikte projedeki zaman döngülerini yönetir.(Koşu, dağıtım, proje)
- Toplantıları yönetir, toplantı tanımlar, bireylerin katılımlarını kontrol eder.
- Efor raporlarını alır, efor girişi yapar.
- Olay yönetimini yapar.
- Proje dosya yönetimini yapar.

Takım Üyesi

Takım üyesinin uygulamada gerçekleştirebileceği işlemler Şekil 4.25'deki diyagram ile gösterilmiştir.



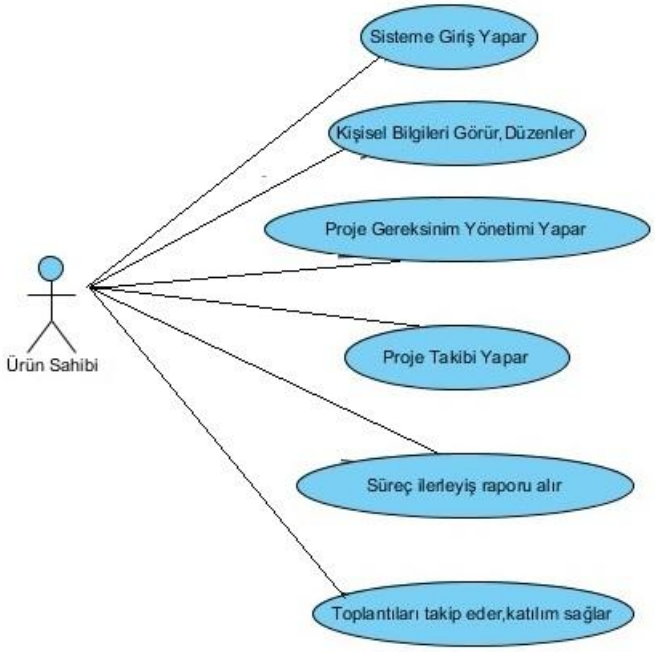
Şekil 4. 25 ScrumMApp'da takım üyesi

Takım sürecin en önemli üyelerinden ve uygulamanın geliştirilmesinden, gereksinimlerin gerçekleştirilmesinden sorumlu bireyler olarak sistem de temel görevleri atandığı projelere katılımda bulunma ve kendi kendine organize olarak görev üstlenme, toplantılara katılma, efor girişi yapmaktır.

- Projede gereksinim girişi yapıp, efor tahmini yapar.
- Toplantı katılımı gerçekleştirir.

Ürün Sahibi

Ürün Sahibi'nin uygulamada gerçekleştirebileceği işlemler Şekil 4.26'daki diyagram ile gösterilmiştir.



Şekil 4. 26 ScrumMApp'da ürün sahibi

Günümüzde Scrum kullanan organizasyonlarda sürecin doğru ilerleyememesinin en büyük sebeplerinden biri yoğunluk veya başka nedenler ile ürünü ve sürecin ilerleyişini takip etmeyen, sürece tam dahil olamayan Ürün Sahibi rolündeki bireylerdir.

ScrumMApp ile bu rolü projeyi izlemesini ve sürece daha aktif katılmasını hedefliyorum.

Bu rol projede doğru ürünün üretilmesinden sorumludur. Gereksinim yönetimi yaparak istediği ürünün ürütülmesini, üretimin gidişatını ve toplantıları takip ederek, sürecin doğru işletilmesine katkı sağlar.

Misafir Kullanıcılar

Scrum, süreç olarak dış katılıma karşı olmayan ve sürece katkısı olabilecek bireyleri de süreç içerisine kabul eden bir süreç modelidir. Bu haliyle bu süreci yönetmeye aday bir yazılım sistemine misafir kullanıcıların kabul edilmemesi kabul edilemez. Ki bu insanlar olay veya görev ile ilgili efor harcıyorlar ise bu görev veya olay detaylarını görebilmeli, proje maliyetinin doğru hesaplanabilmesi için efor girişi yapabilmeleri gerekir. (Şekil 4.27)

Şekil 4. 27 ScrumMApp'da misafir kullanıcı

Sonuçlar

Pilot olarak birkaç yazılım projesinde kullanılan ScrumMApp uygulaması geri beslemeler sonrası geçirdiği düzeltmelerle ile daha başarılı bir hal almıştır.

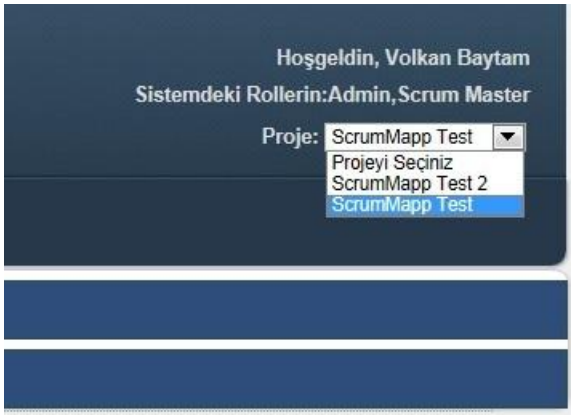
Bir yandan Scrum sürecini yönetip, bir yandan da süreç ilgili bilgiler vererek tüm bireylere süreci tanıtmaya ve öğretmeye çalışan uygulama direk hedef kitleden alınan kullanıcı hikayeleri ve gereksinimler çerçevesinde geliştirildiği için süreç için özel bir süreç yönetim aracı olmuştur.

Uygulamadan alınan ekran görüntüleri aşağıda görüntülenmektedir;

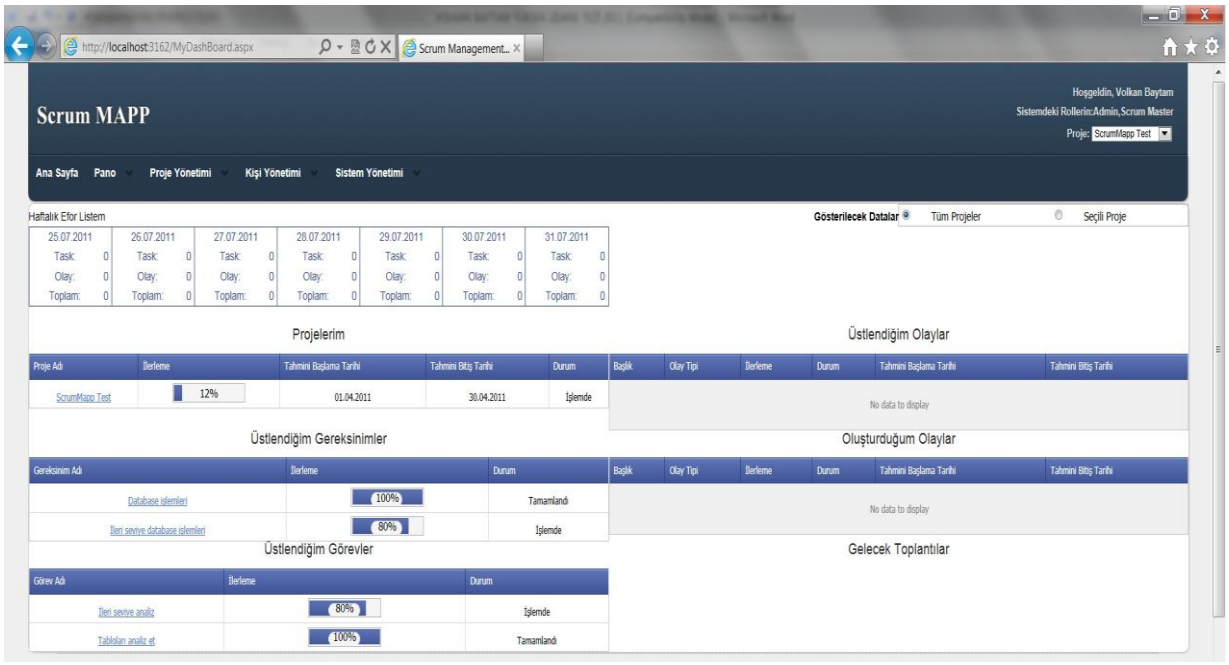
Şekil 4. 28 ScrumMApp giriş ekranı

Şekil 4.29 ScrumMApp kullanıcılara süreci tanıtıcı bilgilerde veriyor

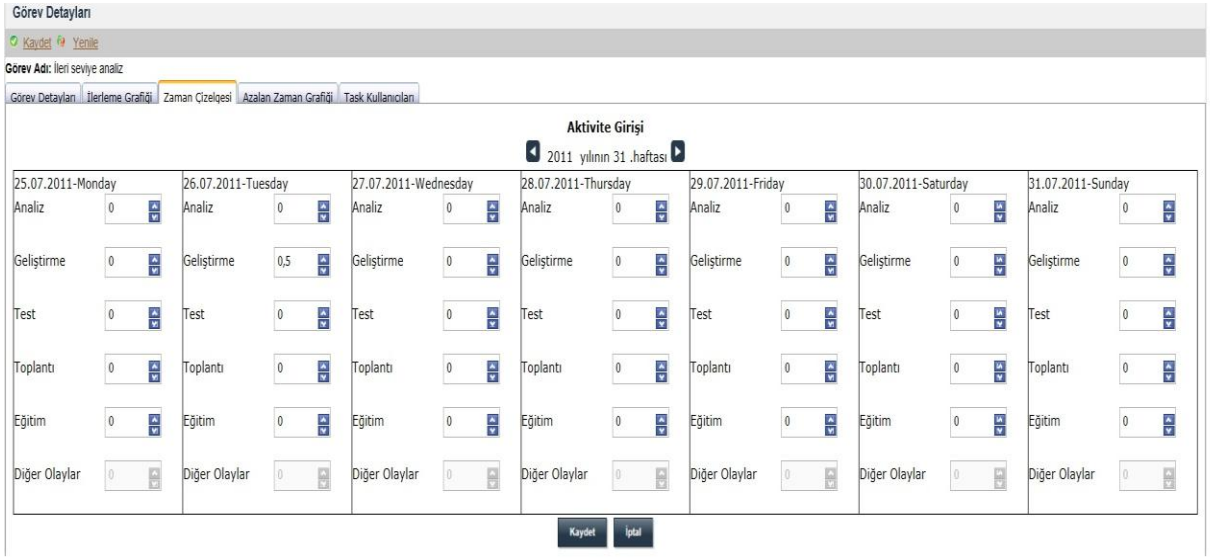
Şekil 4.30 ScrumMApp proje panosu



Şekil 4.31 *ScrumMApp* proje seçimi



Şekil 4.32 ScrumMApp kişisel pano



Şekil 4.33 ScrumMApp efor girişi

BÖLÜM 5

SONUÇ VE ÖNERİLER

Bilgisayar Mühendisliği eğitimimin ilk yıllarından itibaren özellikle yazılım ile alakalı derslerde ağırlıklı olmak üzere hep bir ifade karşıma çıkardı: “Yazılım projeleri büyük oranda başarısız olmaktadır.” Yazılım projelerindeki başarısızlık durumu bu konuda çalışan bireylerin temel sorunuydu, yapılan araştırmalar, yöntemler, metodlar hep bu temel sorunu çözmeye yönelikti.

Bu ifade çalışma hayatı ve okulu beraber yürüttüğüm bu dönemde benimde artık hedef cümlem ve uğraşım olmuştu.

Çalışma hayatında da bu cümlenin teorik bir söylemden ibaret olmadığını gördükten sonra bu çalışma ile problemin temel nedenlerinden birinin yanlış proje yönetim süreci seçimi ve yanlış süreç yönetimi olduğu fikri üzerine giderek organizasyonlara başarılı olma adına öneriler sundum.

Her süreç modelinin her organizasyonda başarılı olması gibi bir zorunluluk yoktur. Organizasyonların yapısı, çalışanların yetkinlikleri, organizasyon gelenekleri, çalışan sayısı, geliştirilen projelerin çeşitleri süreç seçiminde önemli faktörlerdendir.

Ancak yazılım projelerinin genelde başarısızlıkla sonlanması artık süreç ve süreç yönetimlerinde de değişikliklere gidilmesi gerektiğini göstermektedir.

Çevik süreçler bu değişimi arayan organizasyonların karşısına geleneksel yöntemlere göre farklı bir düşünce bir yapısı ile çıkmıştır. Her ne kadar hafızalarda yerleşmiş temel olguları yok edip kişileri çevikleştirmek zor olsada organizasyonlar ciddi bir şekilde çevik süreçlere geçiş yapmaktadır. Çevik süreçlerin arasında yükseliş gösteren Scrum’ın son

zamanda Türkiye'de ki şirketlerinin de favorisi olması sürecin ilerleyen yıllarda daha iyi bir yerde olacağının bir göstergesi olsa gerek.

Scrum süreci işleyişi nedeniyle bir nevi "elbiseyi giyecek kişinin üstünde dikmek" gibi ürünün müşterinin de bulunduğu ortamda onun belirlediği ve istediği şekilde üretilmesini sağlar. Bu hali ile temel sorunlardan biri olan müşteri memnuyeti sağlayamama sorununa çözüm getirmeye aday bir süreçtir.

Ancak sorunun sadece doğru süreç seçimi ile çözülmeyeceği, bunun yeterli olmayacağı fikriyle organizasyonların Scrum'ı doğru işletebilmeleri adına bir yazılım sistemi geliştirmeyi kararlaştırıp, ScrumMApp'i geliştirdim.

Yurtiçindeki Scrum'da deneyimli ve sürece yeni başlayan organizasyondaki bireyler ile yapılan anket ve görüşmeler sonrası geliştirdiğim bu özel süreç işletim yönetim aracıyla gerçekleştirilen pilot uygulamalarda aldığımız başarı sonrası ileri zamanlarda süreç yönetimine olumlu katkıları olacağını konusunda umut vermiştir.

Aldığımız olumlu dönüşler bu uygulamanın temel katkılarının sürecin güncel takibinin sağlanması, sürece yeni bireylere sürecin ve işleyişinin uygulama ile öğretilebilmesi, sürecin doğru kuralları ile işletilmeye zorlanması bu şekilde projenin bir Scrum denemesi değil tam bir Scrum süreci olarak yürütülmesinin sağlanması olmuştur.

Bu geri beslemeler ile bölümün başında bahsettiğim genel soruna karşı ScrumMApp uygulaması ile projelerin başarı oranını artırma yolunda umutlarımı geleceğe taşıyorum.

KAYNAKLAR

[1] ACM Yazılım Çözümleri, "Çevik Yazılım Geliştirme Agile", http://www.acm-software.com/Pdf/AboutAgile.pdf, 11 Nisan 2011.

[2] Sone, S.P., (2008). "Mapping Agile Project Management Practices To Project Management Challenges For Software Development", Dissertation, Faculty Of Argosy University, Washington DC College of Business, Washington.

[3] Schwaber, K., (1995). "Scrum Development Process", http://www.cse.chalmers.se/~feldt/courses/agile/schwaber_1995_scrum_dev_process.pdf, 17 Kasım 2010.

[4] Shenhar, A. ve Dvir, D., (2007). "Reinventing Project Management:The Diamond to Succesful Growth and Innovation", 1 Edition, Harvard Business School Press, Boston.

[5] De Carlo, D., (2004). "Leading and Managing Extreme Projects", Leader to Leader, 2004(34): 51-58.

[6] Schwaber, K., (2004). "Agile Project Management With Scrum", 1 Edition, Microsoft Press, Washington.

[7] Şenkaya, E., (2009). "Yazılım Projelerinde Başarı Anahtarları", CIO CLUB Bilişim Dergisi, Haziran: 54-57.

[8] Cobb, M., (1995). "Unfinished Voyages A Follow-Up to The CHAOS Report", http://www.ics-support.com/download/StandishGroup_CHAOSReport.pdf, 7 Ocak 2011.

[9] Shine Technologies, (2003). "Shine Technologies Agile Methodologies Survey Results", http://www.shinetech.com/attachments/104_ShineTechAgileSurvey2003-01-17.pdf, 20 Ocak 2011.

[10] Wikipedia. "Unified Process", http://en.wikipedia.org/wiki/Unified_Process, 20 Ekim 2011 .

[11] Rational Software Corporation, (2001). "Rational Unified Process: Best practices for software development teams", TP026B(11/01), 20 Ekim 2011.

[12] Agilemanifesto.org, ”Manifesto for Agile Software Development”, http://agilemanifesto.org, 22.Mart 2011.

[13] Cristal, M., Wildt, D. ve Prikladnicki, R., (2008). “Usage of SCRUM Practices within a Global Company”, IEEE International Conference on Global Software Engineering, 17-20 August 2008, Bangalore.

[14] Highsmith, J. ve Cockburn, A., (2001). “Agile Software Development: The Business of Innovation”, IEEE Computer, 34(9): 120-122.

[15] Highsmith, J., (2002). “What is Agile Software Development?” CrossTalk The Journal of Defense Software Engineering October 2002: 4-9.

[16] Augustine, S., (2005). “Managing Agile Projects”, 1 Edition, Prentice Hall, New Jersey.

[17] Highsmith, J., (2004). “Agile Project Management: Creating Innovative Products“, 1 Edition, Addison-Wesley Professional, Boston.

[18] Beck, K., (1999). ”Extreme Programming Explained:Embrace Change”, IEEE Computer, 32(10): 70-77.

[19] Cockburn, A. ve Highsmith, J., (2001). “Agile Software Development: The People Factor”, IEEE Computer, 34(11): 131-133.

[20] Çatal, Ç., (2005). “RUP ve XP Süreçlerinin Karşılaştırılması, http://www.csharpnedir.com/articles/read/?id=458, 20 Ekim 2011

[21] VersionOne, “Agile Survey” http://www.versionone.com/state_of_agile_development_survey/10, 15 Şubat 2011.

[22] Boehm, B., (2002). “Get Ready for Agile Methods, with Care”, IEEE Computer, 35(1): 64-69.

[23] Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L. ve Zelkowitz, M., (2002). “Empirical Findings in Agile Methods”, XP/Agile Universe 2002: 197-207.

[24] Cohen, D., Lindvall, M. ve Costa, P., (2003). "Agile Software Development", The Data & Analysis Center for Software (DACS) Tech Report Number: DACS-SOAR-11.

[25] Karlıdere, T. ve Kalıpsız O., (2003). “Yazılım Mühendisliği Projelerinde Çevik Yaklaşımların Yeri” EMO 1. UYMS, 23-25 Ekim 2003, İzmir.

[26] Murru, O., Deias, R. ve Mugheddu, G., (2003). “Assessing XP at a European Internet Company”, IEEE Software, 20(3): 37-43.

[27] Laurie, W. ve Alistair, C., (2003). “Guest Editors' Introduction: Agile Software Development: It's about Feedback and Change”, IEEE Computer, 36(6): 39-43

[28] Sutherland, J. ve Ph.D. Schwaber K., (2010). ”Scrum Guide”, http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf, 15 Ekim 2010.

[29] Scrum For Team System, “Scrum For Team System Scrum Introduction”, http://www.scrumforteamsystem.co.uk/, 17 Mayıs 2011.

[30] Takeuchi, H. ve Nonaka, I., (1986). "The New Product Development Game". Harvard Business Review January February 86: 2-11

[31] Sutherland, J., (2004). "Agile Development: Lessons learned from the first Scrum", http://www.scrumalliance.org/resources/35, 14 Ekim 2010.

[32] Sutherland, J. ve Schwaber, K., (1995). "Business object design and implementation", OOPSLA 1995, 15-19 October 1995, Texas.

[33] Hundermark, P., (2009). "Do Better Scrum", http://www.scrumsense.com/wp-content/uploads/2009/12/DoBetterScrum-v2.pdf, 20 Ocak 2011.

[34] Judy, K. Krumins-Beens, I., (2008). "Great Scrums Need Great Product Owners: Unbounded Collaboration and Collective Product Ownership", HICSS 2008, 07-10 January, Hawaii.

[35] Sutherland, J. ve Ph.D. Schwaber, K., (2007). "The Scrum Papers:Nuts, Bolts and Origins of an Agile Process", http://www.crisp.se/scrum/books/ScrumPapers20070424.pdf, 13 Eylül 2010.

[36] Välimäki, A. ve Kääriäinen J., (2008). "Patterns for Distributed Scrum – A Case Study", IESA 2008: 85-97.

[37] Cohn, M., (2003). "Toward a Catalog of Scrum Smells", http://www.mountaingoatsoftware.com/system/article/file/11/ScrumSmells.pdf, 21 Mart 2011.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı: Volkan BAYTAM

Doğum Tarihi ve Yeri: 26.03.1987 Kayseri

Yabancı Dili: İngilizce

E-posta: vbaytam@gmail.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Müh.	Erciyes Üniversitesi	2008
Lise	Fen Bilgisi	Kayseri Sümer Lisesi	2004

İŞ TECRÜBESİ

Yıl	Firma/Kurum	Görevi
2010-	Softtech-Türkiye İş Bankası	Yazılım Mühendisi
2008-2010	Itr Bilişim	Yazılım Mühendisi

YAYINLARI

Bildiri

1. Volkan Baytam, Oya Kalıpsız “Scrum Yazılım Geliştirme Modeli Yönetim Aracı: ScrumMApp” V. Ulusal Yazılım Mühendisliği Sempozuyumu Bildiri Kitabı 2011