KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YÜKSEK LİSANS TEZİ

OYUN PROGRAMLAMADA ÇEVİK YÖNTEMLER

ŞAHİN MERCAN

KOCAELİ 2021

KOCAELİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ

ANABİLİM DALI

YÜKSEK LİSANS TEZİ

OYUN PROGRAMLAMADA ÇEVİK YÖNTEMLER

ŞAHİN MERCAN

Prof.Dr. Yaşar BECERİKLİ		
Danışman,	**Kocaeli Üniversitesi**	**.....**
Prof.Dr. Nejat YUMUŞAK		
Jüri Üyesi,	**Sakarya Üniversitesi**	**.....**
Dr.Ögr. Üyesi Alev MUTLU		
Jüri Üyesi,	**Kocaeli Üniversitesi**	**.....**

Tezin Savunulduğu Tarih: 26.01.2021

ÖNSÖZ VE TEŞEKKÜR

Bu tez çalışmasında yazılım şirketlerinin son yıllarda sıklıkla kullandığı çevik yöntemler konusu ele alınmıştır. Çevik yöntemlerin tarihsel gelişim süreci ve projelerde nasıl kullanıldığı, avantajları ve dezavantajları araştırılmıştır. Bu araştırmalardan yola çıkarak çevik yöntemlerin proje yönetimi konusunda daha başarılı bir şekilde nasıl yapılabileceği hakkında öneriler sunmaktayız. Bu öneriler ışığında yapılan bir oyun programlama ile ortaya çıkan sonuçlar incelenmektedir.

Tez çalışmam boyunca bana yol gösteren ve tecrübesiyle benim bu çalışmayı bitirmemde bana yardım eden tez hocam sayın Prof.Dr. Yaşar BECERİKLİ ’ye çok teşekkür ederim. Son olarak hayatımın her döneminde bana güvenen ve benim yanımda olan aileme sonsuz teşekkür ederim.

Ocak – 2021 Şahin MERCAN

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Kısaltmalar

AG : Arık Geliştirme
UYG : Uyarlanabilir Yazılım Geliştirme
EP : Ekstrem Programlama
ÖGG : Özellik Güdümlü Geliştirme
AKKG : Açık Kaynak Kodlu Geliştirme
DSG : Dinamik Sistem Geliştirme

OYUN PROGRAMLAMADA ÇEVİK YÖNTEMLER

ÖZET

Son dönemlerde oyun programlamada yaşanan sorunlardan birisi oyun programlama gereksinimlerinin sürekli değişmesidir. Bu yüzden oyun programlarken yürüttüğümüz sürecin bu değişime ayak uyduramaması geliştirdiğimiz ürünün başarısızlıkla ya da daha yüksek maliyetlerle sonuçlanmasına yol açmaktadır.

Bu nedenle son zamanlarda değişen ortam koşullarına karşı daha dinamik ve modern bir çözüm getiren çevik yöntemlerin proje yönetiminde uygulaması giderek artmaktadır.

Tezimizde ilk olarak çevik süreçlerin ortaya çıkışı, temel ilkeleri ve bu konu hakkındaki diğer çalışmalardan bahsedilecektir.

İlerleyen bölümlerde ise çevik programlama yöntemi olan Scrum metodu kullanılarak "Huysuz Top" ve "Balon Vurma" uygulamaları yapılacaktır. Çıkan sonuçlar ışığında çevik metodolojinin dezavantajlarından biri olan proje teslim tarihinin nasıl tespit edilebileceğine yönelik bir çalışmadır.

Anahtar Kelimeler: Çevik Yöntemler, Oyun Programlama Süreci, Oyun Proje Yönetimi, Scrum.

AGILE METHODS IN GAME PROGRAMMING

ABSTRACT

One of the problems experienced in game programming lately is that game programming requirements are constantly changing. Therefore, the inability of the process we carry out while programming the game to keep up with this change causes the product we develop to fail or to result in higher costs.

Therefore, the use of agile programming, which brings a more dynamic and modern solution to changing environment conditions, is increasing day by day and the project teams are transitioning to these processes.

In this thesis, firstly historical development, content, rules and methods of Agile methods will be presented with literature researches.

Using the Scrum method, which is the agile programming method, in the next section. "Grumpy Ball" and "Balloon Shooting" applications will be made. In the light of the results, it is a study on how to determine the project deadline, which is one of the disadvantages of agile methodology.

Keywords: Agile Methods, Game Programming Process, Game Project Management, Scrum.

GİRİŞ

Yazılım süreçlerine başlarken ilerleyen safhalarda değişim isteklerin oluşabileceği öngörülmektedir. Bu nedenle geleneksel yöntemler yazılım süreçlerinin en başında kapsamlı bir çalışma yaparak oluşabilecek ihtiyaçların hepsini belirleyip ilerleyen safhalarda oluşabilecek değişiklikleri önlemeyi esas almaktadır. Ancak hızla değişen, gelişen ortam ve piyasa koşullarına karşın artık proje gereksinimleri daha fazla ve hızlı bir şekilde değişmektedir.

Proje başında bütün ihtiyaçların eksiksiz bir şekilde belirlenmesi neredeyse imkânsız bir hal almaktadır. Bu nedenle kullandığımız proje yönetim modelinin bu şartlara olabildiğince uyum sağlaması beklenmektedir. Uyum sağlama konusunda yeni teknikler ortaya süren çevik yöntemlerin kullanıldığı proje yönetim biçimleri ortaya çıkmıştır [1, 2]. Bu yöntemler yapılan uygulamaya göre esneklik gösterebilir, istenirse tümünde istenirse belirli bir kısmına uygulanabilir ayrıca geliştirmeyi yapan takıma uyacak şekilde kullanılabilir [3, 4, 5].

Çevik metot proje yönetimi çeşitleri ise Arık Geliştirme (AG), Uyarlanabilir Yazılım Geliştirme (UYG), Scrum, Ekstrem Programlama (EP), Kristal Yöntemleri, Özellik Güdümlü Geliştirme (ÖGG), Açık Kaynak Kod Geliştirme (AKKG) ve Dinamik Sistem Geliştirme (DSG) yöntemleridir. Belirtilen metotlardan Arık Geliştirme dışında diğerleri 2001 yılında Çevik Yazılım Geliştirme Manifestosu oluşturulduktan sonra çevik yazılım ilkeleri tanımlanmaya başlanmıştır. Bu ilkeler aşağıda verilmiştir.

1. “Süreç ve geliştirme araçlarının yerine bireyler ve bireyler arası ilişkileri” [6]
2. “Detaylı dokümantasyon yerine koşan yazılımı” [6]
3. “Sözleşme görüşmeleri yerine müşteri iş birliğini” [6]
4. “Plan takip etmek yerine değişikliklere ayak uydurmayı” [6]

Belirtilen 4 madde bu yöntemlerin ortaya çıktığı bildiride bulunmakta ve bu metotların etkili bir şekilde kullanılması için gerekli olan maddelerdir [6].

Yukarıda belirttiğimiz çevik yöntem metodu çeşitleri birbirlerine benzese de bunları ayıran yönleri de bulunmaktadır [7]. Bu yöntemlerin birkaçı uyumlu biçimde çalışmayı hedeflerken birkaçı da projeyi geliştirmeye odaklanmıştır.

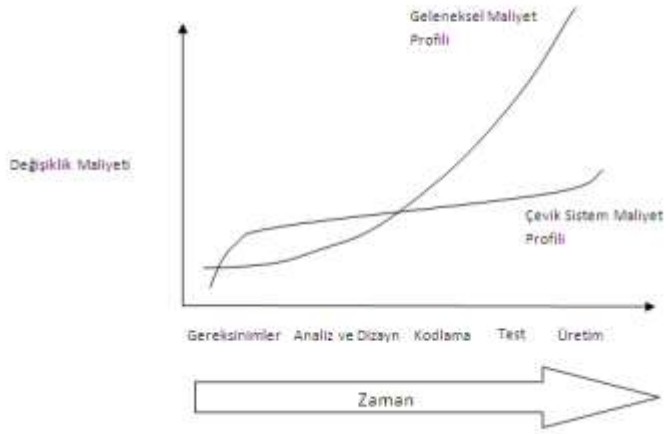
Bu metotların güçlü noktaları, yazılım geliştirme sürecinin değişen gereksinimlere karşı daha duyarlı hale getirilebilmesidir [8]. Bu metotlarda, çalışan yazılım ayrıntılı dokümantasyonun daha üzerinde tutulur; kişiler ve etkileşimler, araçlar ve süreçlerden daha önemli kabul edilir ve müşteri iş birliği müzakere edilen sözleşmeden daha değerlidir. [9]

1. LİTERATÜR ÖZETİ

1.1. Çevik Yazılım Geliştirme Anketleri

Bu alanda yapılan araştırmalar ışığında projelerin başarı oranları oldukça düşüktür.

Şekil 1.1'de geliştirmelerdeki değişikliklerin geleneksel ve çevik süreçlere göre dağılımı gösterilmiştir. Proje yönetim biçiminin esnek olmasının önemi açıkça görülmektedir.



Şekil 1.1. Değişiklik maliyeti-zaman grafiği

Tablo 1.1. Chaos anketine göre proje başarı oranları

Yıl	Başarılı (%)	Ek Maliyet (%)	Başarısız (%)
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

Yukarıda belirtilen sonuçlar oldukça dikkat çekicidir. Chaos'un araştırmayı yaptığı bütün dönemlerde başarı oranında sürekli artış gözlenirken ilk defa 2009 yılında başarı oranının düştüğü gözleniyor [10].

2009 yılındaki diğer bir sonuç ise oldukça yüksek bir oranda projenin ek maliyet ile tamamlandığı gözlemlenmiştir [10].

Son olarak da %24'lük bölüm ise başarısız bir şekilde sonuçlanmamış ve proje tamamlanamamıştır [10].

Bu sonuçlara göre projelerin başarısızlık yüzdesinin yükseldiği görülmektedir. Başarı yüzdesi olarak da oldukça düşük seviyelerde olduğu gözlemlenmiştir. Buradan çıkan sonuçlardan anlaşıldığı üzere başarılı şekilde projeyi tamamlamak oldukça zor olduğu gözleniyor [10].

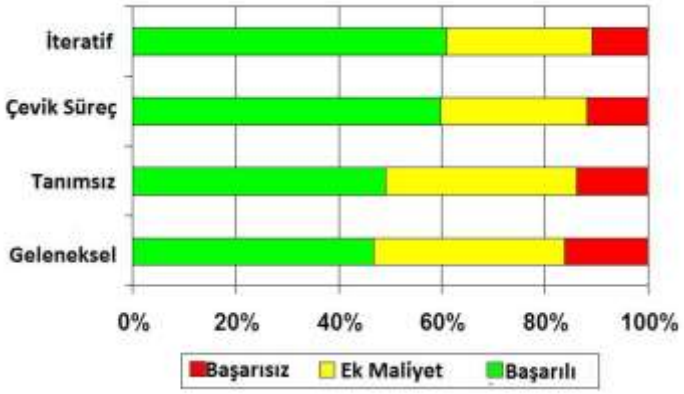
Aşağıdaki tabloda ise aynı şirketin yaptığı tamamlanmış projelerdeki başarı ölçütlerin hesaplanmaya çalışılmıştır [11]. Tablo 1.1.'da başarılı olma kriterlerini görebiliriz.

Tablo 1.2. Chaos 1995 anketine göre geliştirmenin Başarı ölçütleri

Başarı Ölçütleri	Puan
Hedef kitle katılımı	19
Yönetim desteği	61
İhtiyaçların tam belirtilmesi	15
Doğru planlama	11
Reel beklentiler	10
Daha küçük geliştirme bölümleri	9
Uzman geliştiriciler	8
İşi sahiplenme	6
Açık vizyon, hedefler	3
Konsantre takım	3
Sonuç	100

Shine Teknoloji 2003 araştırması verilerinde ise çevik metotların uygulandığı geliştirmeler geleneksel metotların uygulandığı geliştirmelere oranla verimliliğin %93 oranında yükseldiği ortaya koyulmuştur [12].

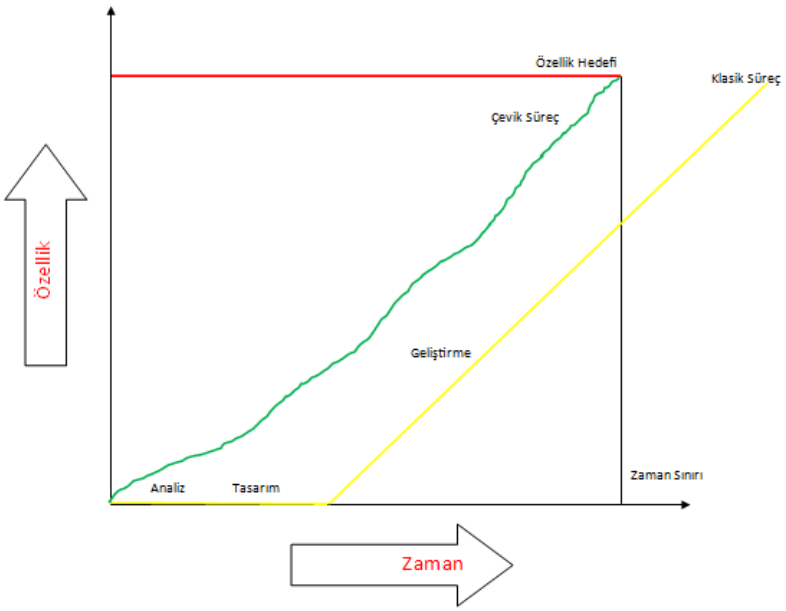
Bu çalışmalardan görüleceği üzere geleneksel süreçler artık başarılı olma yolunda iyi bir araç değildir. Kurumların ve kişilerin karakterleri, beklentileri ve yazılım geliştirme süreci çevik yöntemlere daha yatkındır. Bu durumu çevik süreç kullanan organizasyonların başarılı proje oranlarından görebiliriz.



Şekil 1.2. Scott W. Ambler'in anketine göre proje başarı oranları

Yukarıda araştırmada metotların başarılı olma oranları Şekil 1.3.'de gösterildiği şekildedir;

Ayrıca geleneksel yöntem ile çevik yöntemlerin zaman sapmasıyla ilgili grafiği Şekil 1.4.'da gösterilmiştir. Bu şekilden anlaşılacağı üzere geleneksel yöntem sürece biraz daha geç başlamaktadır. Ayrıca çevik sürecin dinamik yapısı zaman sapması üzerinde herhangi bir olumsuz sonuç doğurmadığı gözlenmiştir.



Şekil 1.3. Geleneksel ve çevik modellerin karşılaştırılması

1.2. Çevik Yazılım Geliştirme Özellikleri

Bu yöntemin en önemli özelliği projedeki kişilere, ekipmanlar ve dokümanlardan daha fazla önem verilmesidir. [13]. Bunun nedeni geliştirmeyi yapan yazılımcılar olduğu için ayrıca geliştirmedeki ekipmanları da kullanan yazılımcılar olduğu için kişilere daha çok önem vermektedir. [14]. Bu nedenle bu metotlar geliştiren kişilerin özgüvenini arttırmayı hedefler ve bu nedenle kişilere her türlü desteği vermeye önem gösterir [14].

Çevik metotlar geleneksel metotların aksine kişilere önem verdiği için diğer ekipman belge vb. unsurları çok fazla dikkate almadan bir an önce geliştirmeye başlamayı hedeflemektedir. Bu sebeple geliştirme başlarken uğraşılan doküman gibi benzeri şeyleri sadece istenilen durumlarda hazırlayarak daha hızlı ve esnek ilerlemeyi amaçlar [15].

Çevik metotları diğerlerinden ayıran husus ise değişen şartlara her şeyden daha çok değer vermesidir [15]. Geliştirici ekip müşteri beklentilerini karşılamaya çalıştığı için daha kısa sürede işin tamamlanacağı sonucuna varılabilir. Fakat ileride müşterinin beklentileri genel olarak farklılaştığından geliştirme başarısız da olabilir. Sonuç olarak geliştirme başında direk sonuca odaklanarak müşterinin daha sonraki aşamalardaki beklentilerini dikkate almayarak ya da bu beklentileri olumsuz bir şekilde karşılamak geliştirmenin olumsuz sonuçlanmasına yol açar [15]. Çevik metotlar ise bu durumları önleyecek şekilde sürekli esnek bir geliştirme sürecini temel almaktadır.

Çevik metotlar, dinamik, bağlama özgü, ısrarla değişmeyi zorlayan ve büyüme odaklı bir yazılım geliştirme metodudur. Bu durum metodun kullanımındaki başarma ve kazanma isteği ile ilgilidir. Günümüzde, ortaya çıkan rekabetçi alanlarda başarılı olma, pazar payı ve müşteriler yazılım geliştiren şirketlerin odak noktası haline gelmiştir [16].

Geliştirme ekibini verimli kullanmak manevra kabiliyeti, hız ve maliyet tasarrufu kazandırır [16]. Örneğin birlikte çalışan birkaç tasarımcı, her birinin tek başına üreteceği tasarımdan daha iyi bir tasarım ürünü ortaya koyabilir [16].

Bireyler arası etkileşimlere bağlı olarak hızlı bilgi paylaşımı, gerektiğinde süreci değiştirmeyi kolaylaştırır [16]. Çalışan yazılımı kullanmak bizim ne kadar hızlı sonuç üretebileceğimizi ve geribildirim alabileceğimizi ölçmemize izin verir [16]. Bireyler arasındaki sıkı etkileşim, belgelemeyi en aza indirgeme durumunun olumsuz etkilerini telafi eder [16].

Bu özellikleri aşağıdaki gibi sınıflayabiliriz.

1.2.1. Odaklanma ve perspektif

Çevik yöntemler proje perspektifine sahiptir. Çevik yöntemlerin odak noktası proje ve proje ekibidir [17].

1.2.2. Yönetim

Çevik metotlarda yönetim geliştirme yapan takımın önüne çıkabilecek sorunları gidermeyi hedefleyerek takımın geliştirme yaparken takılmadan ilerleyebilmesini amaçlar. [17].

1.2.3. Planlama

Çevik metotlar sürekli değişen bir planlama yapısı ortaya koyar. Bunlar kısa süreli olanı koşu planlama şeklindedir. Diğeri ise koşuların planlandığı genel bir planlama yapılması şeklindedir [17].

1.2.4. Öğrenme

Çevik metotlar geliştirme yapılırken o anda öğrenmeyi hedefleyen bir sistemdir. Sorunun karşılaşıldığı anda araştırma yapılarak çözülmesi hedeflenir. [17].

1.2.5. Değerlendirme

Çevik metotlarda değerlendirme ortaya çıkan proje sonucunda müşteriden alınan geri dönüşler üzerinden belirlenir [17].

1.2.6. Kişisel gelişim

Çevik metotlar kişilere önem verdiği için geliştirme konusunda deneyimli insanlardan meydana gelmektedir [17].

1.2.7. Proje yaşam döngüsü faaliyetleri

Çevik metotlarda proje geliştirme aşaması ile uygulama testi aynı anda yapılması istenmektedir. Bu nedenle ek maliyet veya başarısız durumların olabilecek en erken şekilde tespit edilebileceği yinelemeli bir yol izlenmektedir [17].

1.2.8. Tahminleme

Bu kısımda ekibin uzman olması beklendiğinden ötürü ilk başta belirli süreler verilmektedir. Daha sonra koşular geçtikçe başta verilen süreler ile uygun şekilde ilerlemesi öngörülmektedir. Buradaki beklenen durum geliştirme ilerledikçe ek bir süreye ihtiyaç duyma oranını minimumda tutmaktır [17].

1.3. Çevik Yazılım Geliştirme Mimarisi

Yazılım mimarisi, uygulamanın çevikliğini sürdürmek ve var olan uygulamanın fonksiyonel ve fonksiyonel olmayan gereklilikleri gerçekleştirebilecek bir yapıya sahip olduğundan emin olmak için kendi geliştirme aktivitesini kontrol etmek üzere, takım tarafından kendine dayatılan kısıtların kümesidir [18]. Diğer deyişle yazılım mimarisi, yazılım sisteminin yapısını ve karakteristiğini tanımlayan kısıtlar kümesidir. [18]

Bir uygulamanın sürdürülebilirliğini (ve dolayısıyla çevikliğini) arttıran ve uygulamaya düzen getiren iki şey vardır. Bunlar [18];

a) Yazılım Mimarisi (Makro Seviyede)
b) Kodlama Standartları (Mikro Seviyede)

Yazılım mimarisi, genellikle uygulama seviyesinde tanımlanır ve tek başına çevik olmalıdır [18]. Ancak kodlama standartları kodu daha okunabilir yapar ve bu yüzden müşteri/takım düzeyinde veya organizasyon seviyesinde tanımlanmalı ve yazılım yaşam döngüsü içerisinde hiçbir zaman değiştirilmemelidir [18].

Çevik yazılım geliştirme manifestosunun temel prensiplerinden birisi, en iyi mimari ve tasarımın ortaya çıkması için kendi kendine organize olan ekiplerin olması gerekliliğidir [18]. Bu nedenle, çevik proje mimarisi bir kişi tarafından veya bir anda tanımlanmış olmamalıdır. Bu mimari, birbirini denetleyen takımlar tarafından tanımlanmalı ve zamanla en iyi mimari haline gelmelidir [18].

1.4. Çevik Yöntemlerin Yazılım Projelerine Uygunluk Kriterleri

Öncelikle proje yönetim metodundan beklenen husus geliştirmeyi başarılı bir şekilde tamamlamasıdır [19, 20]. Ancak yine de metodu seçerken geliştirmenin uyumluluğu kontrol edilmelidir [20]. Yanlış bir metot seçme geliştirmenin olumsuz sonuçlanmasına yol açmaktadır [20]. Bu yüzden hangi metotların ne zaman kullanılacağının bilinmesi önemlidir.

İncelediğimiz bir makalede bu araştırma yapılmış ve şu sonuçlara ulaşılmıştır. Ekibin büyüklüğü, ekibin deneyimi, müşteri profili, uygulama kritikliği, bakım safhası ve çevikliğe yatkınlık bölümleri geliştirmeye bakılarak karar verilmelidir [20].

1.4.1. Ekibin büyüklüğü

Bu konunun önemi çevik metotlarda geliştirici ekipten birebir iletişim kurması beklenmektedir. Bu nedenle birebir iletişim kurmayı belirleyen can alıcı hususta biride takımdaki kişi sayısıdır. Genel olarak takımdaki işi yapan çalışan sayısı 15'den fazla değildir [20, 21]. Eğer takımımız bu sayıdan daha fazla ise bunları daha küçük takımlara bölerek geliştirmeye devam edilmelidir. Aksi taktirde iletişim kurma kötü yönde etkilenecektir.

Çevik metotlarda takımın birbiriyle iletişimde olması gerektiği gibi müşteri ile iletişim de oldukça önemlidir [21]. Bu yüzden birebir iletişim kuracak kişilerin aynı mekanlarda geliştirme yapması gerekmektedir. Bu hususta çeşitli iletişim uygulamaları kullanılmaya çalışılsa da geliştiricilerin performansını yükseltmenin en önemli yolu bir arada olmalarından geçmektedir [21, 22].

1.4.2. Ekibin deneyimi

Bu konuda takımın uzman kişilerden olması beklenmektedir. Bunun sebebi olarak takım direk olarak geliştirmeye başladığı için yapılacak geliştirmeyi bilmesi beklenmektedir [23]. Burada en azından takımın tamamının değilse bile bazılarının bu tarz geliştirmelerde tecrübe sahibi olması beklenir [23]. Bu tip durumlarda bilen kişilerle bilmeyen kişiler sürekli olarak değişimli bir şekilde geliştirme yapması gerekmektedir [24]. Böyle bir geliştirmeyi yapabilmek için takımdaki bilen kişilerin diğer kişilere öğretmeyi hedeflemesi gerekmektedir [24].

1.4.3. Müşteri profili

Bu yöntemlerde müşteri ile aynı anda iletişim halinde olması beklenir [24]. Çünkü direk olarak geliştirmeye başlanır ve müşterinin istekleri doğrultusunda gerekli düzenlemelerin zamanında yetiştirebilmesi için müşterinin de proje ekibiyle bire bir sürece dahil olması gerekir. [24]. Eğer müşteri beraber geliştirmeye yatkın olmadığı durumlarda geliştirmeyi üstlenen başka bir şirket ile geliştirme yapmak [25] durumunda kalınırsa geliştirme kötü yönde etkilenecektir.

1.4.4. Kritik uygulamalar

Burada kastedilen maliyeti yüksek uygulamalardır. Bu yöntemler için böyle bir geliştirme ile henüz karşılaşılmamıştır [25]. Bu tip geliştirmelerin test maliyeti oldukça fazladır [25]. Bu yöntemin yinelemeli olması sebebiyle istenilen sayıda test yapılması gerekir [25]. Fakat böyle uygulamalarda yeteri sıklıkta test yapılamayacağı aşikâr olup ayrıca geliştirmenin tamamının test edilmesi beklenebilir [25]. Bu yöntem bu tip kritik yani herhangi bir hata olmaması gereken geliştirmelerde yorum yapabilmek için gerektiği kadar fazla örnek olması beklenmektedir [26].

Tam olarak net bir sonuç öngörülemediği için çevik ve geleneksel metotların birlikte kullanıldığı bir metot tercih edilebilir [27].

1.4.5. Bakım safhası

Bakım safhası geliştirmelerde bazen daha kritik olabilmektedir. Geliştirme bittikten sonra oluşabilecek bakım geliştirmeleri proje başlangıcında herhangi bir hazırlık

yapılmadığı için daha maliyetli olmasına yol açabilir. Çevik yöntemlerde direk geliştirmeye geçildiği için bu kısımdaki risk göze alınarak devam edilir [27]. Riski en aza indirgemek için proje bittikten sonra bir doküman oluşturulmalıdır [27].

Bu yöntemlerin bu aşamayı ne derece sağlıklı şekilde geçirebileceği tam olarak öngörülemediği için dikkate alınmıyor [27]. Fakat bu konu hakkında yeteri kadar araştırma yapılmaz ise ileride daha kritik durumlara yol açabilir.

1.4.6. Çevikliğe yatkınlık

Bu yöntemi geliştirmelerde kullanmak için öncelikle yukarıda belirtilen maddelere uygun bir şirket yapısı oluşturması gerekmektedir [28]. Bu yöntemde projenin geliştirilmesindeki hemen hemen bütün kararlara geliştiren ekip karar verdiği için şirketinde bu hususa ayak uydurması yani bu yöntemi benimsemiş olması beklenmektedir [28].

Bazı durumlarda geliştirme bu yönteme uyum sağlayabilir. Fakat geliştiren takıma yeteri kadar sorumluluk yani yetki verilmez ise takım geliştirmenin tamamlanmasında başarısız olacaktır. En azından başarılı olma ihtimali düşecektir.

Ayrıca bu konuda müşteride geliştirme başında herhangi bir iş yapılmadan direk geliştirmeye başlanılmasını istemeyebilir. Bu sebeple müşterinin de bu metoda yatkın olması beklenmektedir. [28].

1.5. Çevik Yazılım Geliştirme Metotları

Yukarıda belirttiğimiz çevik yöntem metodu çeşitleri birbirlerine benzese de bunları ayıran yönleri de bulunmaktadır [28]. Bu yöntemlerin birkaçı uyumlu biçimde çalışmayı hedeflerken birkaçı da projeyi geliştirmeye odaklanmıştır. Bu yöntemin metotları aşağıdaki bölümde aktarılmaktadır.

1.5.1. Ekstrem programlama

Ekstrem programlama geleneksel geliştirme modellerindeki uzun geliştirme döngülerinin neden olduğu problemlere karşı çözüm olarak geliştirilmiştir [29].

(a) “Basitlik, haberleşme, geribildirim ve cesaret temelleri üzerine kurulmuş olan” [29]

(b) “Test yapma, kodlama, dinleme ve tasarım yapma aktivitelerinden oluşan” [29]
(c) “Hızlı geri bildirim, değişimi benimseyen, artımlı değişim ve kaliteli iş prensipleri ile çalışan” [29]
(d) “Aktiviteleri başarıyla gerçekleştirmek için kullanılan 12 pratik içeren (planlama, küçük içerikli sürümler, benzetme, basit tasarım, yeniden düzenleme, önce test geliştirme, ikili programlama, kolektif ortaklık, sürekli entegrasyon, haftada 40 saat çalışma, yerinde müşteri ve kodlama standartları)” [29]
(e) “Gerçek hayatta yukarıda belirtilen uygulamaları başarıyla gerçekleştirebilmek için tanımlanmış stratejilerden oluşan bir yazılım geliştirme yöntemidir” [30, 31].

Ekstrem Programlama (EP) yazılım geliştirme metoduna ilk olarak projenin aşamalara bölünmesiyle başlar. Daha sonra her aşama için müşteriyle beraber gerçekleştirilen toplantılarla ihtiyaçlara karar verilir ve program oluşturulur. Geliştirme aşamasında geliştiriciler, müşteriyle sürekli iletişim halindedirler. Geliştirme aşamasında geliştiriciler planlanan gereksinimlerin yapılmasına konsantre olur ayrıca gerektiğinde geliştirmeyi sadeleştirmek amacıyla proje görsellerinde değişikliğe gidilebilir. Geliştirme aşamasının her safhasında üretilen ara ürünlerin kontrolü yapılır. Geliştirme ilerleme takibi ise müşteri, ekip arkadaşları ve kontrol süreci doğrultusunda genelde gün başlangıcında gerçekleştirilen değerlendirme toplantılarıyla izlenmektedir. Geliştirme ekibi, geliştirmeyi en kısa sürede çalışır duruma getirdiğinde, müşteriden gönderilen revizyonlara göre değişiklikler yeniden planlanıp yapılır [32]. İhtiyaçların geliştirme başında isabetli bir şekilde tanımlanamadığı, ihtiyaçların sürekli değiştiği az zamanda çalışan ürünü meydana getirmeyi hedefleyen işlerde uygulanabilir [32].

1.5.2. Kristal

Bu yöntem değişik alandaki uygulamalara aynı metotlar uygulanamayacağı mantığıyla meydana gelmiştir [33]. Bu yöntem uygulamalardaki ekibin büyüklüğüne ve hataların oluşturduğu sonuçlara yoğunlaşır. Bu yöntem Konfor (Comfort), isteğe Bağlı Kapital (Discretionary Money), Gerekli Kapital (Essential Money) ve Yaşam Noktası (Life) olmak üzere 4 farklı seviye içermektedir [33, 34]. Crystal Clear altı çalışandan oluşan takımlar, Crystal Orange en az on en fazla kırk çalışandan oluşan takımlardan oluşur ve maksimum iki yıllık uygulamalar yapılır [34]. Crystal Orange/Web’de web ile ilgili

uygulamalar tercih edilir [34]. Kristal geliştirmeye konsantre olan, birebir konuşma odaklı ekibi ön planda tutar. Aşağıdaki hususlar bu yöntemin önemli noktalarıdır [35]:

(a) "Her proje farklı birtakım kural, uyum ya da yönteme ihtiyaç duyar."
(b) "Proje çalışmaları insan ile ilgili konularda hassastır ve bu konuların gelişimine bağlı olarak daha da gelişir; insanlar daha iyi oldukça, grup içi çalışmalar daha da iyi olur."
(c) "Etkin iletişim ve sık yapılan teslimler orta seviye iş ürünlerinde gereksinimi azaltır."

İki önemli noktası bulunur:

(a) "1 ile 3 ay arasında değişen en fazla 4 ay süren artımlarla geliştirme yapılır" [35].
(b) "Her bir artım öncesi ve sonrası başarılar ve başarısızlıkların dile getirildiği fikir seminerleri düzenlenir" [35].

Bu yöntem önem derecesine göre kırmızı, turuncu, sarı ve beyaz renklerden oluşur [35]. Önem derecesi kırmızıdan beyaza doğrudur [35]. Yöntemin önemine göre renk giderek koyulaşır. Bu önem derecesine projenin büyüklüğü ve kritikliğine bakılarak karar verilir. Takım yinelemeli programa ve zamana karar verir. Yinelemeli kısmı geliştirmeyi temel alan, ara ürün ve ilerleme aşamasını hedef alır. Başta verilen sürelere göre ve ilerleme durumuna göre uygulamada ne derece başarılı olduğu izlenir. Her yineleme başlangıcında işlere ağırlık verilerek uygulamaya devam edilir. EP ve Scrum'la birlikte çalışabilir, kritik uygulamalarda kristal metodu kötü sonuçlar doğurabilir, ayrıca büyük çaptaki uygulamalarda kullanılması önerilmez [35].

1.5.3. Açık kaynak geliştirme

Günümüzde internetin yaygınlaşmasıyla açık kaynak kod yazılımı giderek yaygınlaşmaya başlamıştır. Bu yöntem farklı donanım ve farklı durumlarda kullanıcı testi yapılabildiği için hataların bulunması ve farklı çözümlerin geliştirilmesine ışık tutmaktadır. Geliştiriciler, istedikleri yerden geliştirmeleri takip edebilmekte, istedikleri geliştirmeyi yapabilmekte ve bu geliştirmeleri uygulamalarında istedikleri gibi kullanabilmektedirler. Ayrıca kodun gerçek sahibiyle doğrudan iletişim kurup destek alabilirler. Herkes istediği geliştirmeyi görebildiği için zamandan kazanç

sağlanmaktadır ayrıca herkesin kendini uzmanlaştırmasına olanak tanımaktadır. Bu metot sorunun belirlenmesini, diğer geliştiricilere ulaşmasını, geliştirmenin yapılmasını ve ara ürün ile test senaryolarının yapıldığı adımları içinde bulundurur. Bu yöntem ekipte bulunan herkesin gerekli aşamalarda konuya dahil olduğu bir yapıyı içerir. Yani herkes belirli aşamalarda geliştirmeye dahil olur işini yapar ve tekrar konu kendine gelene kadar bekler [35].

1.5.4. Uyarlanabilir yazılım geliştirme

Bu yöntemde müşteri istekleri sürekli farklılık gösterdiğinden gereksinimlerde farklılık olmasını önlemeyi değil bu farklılıklarla beraber geliştirme yapmayı esas alır. Yöntemi uygulayabilmek için yukarıdaki tanıma uygun yapıda ekiplere gerek vardır. Süreçten ziyade iş birliğiyle meydana gelen uygulamaya odaklanmayı tercih etmektedirler. Bunun nedeni oluşabilecek belirsizlerin iş birliğiyle daha efektif öneriler sunulmasını esas almaktadır. Uyarlanabilir yazılım geliştirme (UYG)'de;

(a) "Projenin vizyonunu, verilerini ve tanımlarını içeren ortak dokümantasyon tutulur" [35].
(b) "Oturumlar, müşteri odaklı gruplar, süreç iyileştirme ve geliştirme toplantıları gibi iş birliği teknikleri uygulanır" [35].
(c) "İş birliği tekniklerinin planlanmasından sorumlu bir organizatör bulunur" [35].

UYG öncelikle her zaman gelişme ve iş birliği hususuna önem veren, yinelemeli planlamayı esas alır. UYG'nin temelinde 3 aşama vardır: spekülasyon, iş birliği ve öğrenmedir. Spekülasyon aşamasında uygulama hedefi ve yapımında istenilen özellikler oluşturulur daha sonrada takvim belirlenir. Uygulamanın oluşturulması sırasında müşteri ile takımlar bir araya gelerek uygulamaya karar verir. Burada uygulamada ne yapılacağından çok neler öğretileceği üzerinde durulur. UYG'de ekibin nasıl olması gerektiği konusunda net bir tavır belirlenmemiştir. [35].

1.5.5. Özellik güdümlü geliştirme

Özellik güdümlü geliştirme tüm yazılım geliştirme sürecini kapsamaz. Tasarım ve yapı aşamaları üzerinde yoğunlaşır [35]. Özellik Güdümlü Geliştirme (ÖGG) yinelemeli

adımlardan oluşur. Takvim, önem derecesi, görseller ayrıca geliştirme adımlarında işlev denilen müşterinin isteklerinden yararlanılarak geliştirmeye başlanır [35].

ÖGG sekiz temel model içermektedir:

(a) "Problem alanının incelenmesini, tanımlanmasını ve özelliklerin belirlenmesini içeren Alan Obje Modellemesi,"
(b) "Tanımlanmış bir dizi özellik üzerinden geliştirme ve ilerlemeyi izlemeyi içeren Özelliklere Göre Geliştirme,"
(c) "Bakımı ve devamlılığı kolaylaştırmak için her sınıfın bir geliştiriciye aitliğini belirten Bireysel Kod Sahipliği,"
(d) "Küçük ve dinamik olarak oluşturulan Özellik Takımları,"
(e) "Kontrol listesi tabanlı toplantıları ve kod standartlarını içeren Denetleme,"
(f) "Yeni bir özelliğin eklenebileceği, her an için çalışan bir sistemi garanti eden Düzenli Yapılar",
(g) "Versiyonlama takibini sağlayan Konfigürasyon Yönetimi,"
(h) "Tüm önemli organizasyonel yapıya işin tamamı hakkındaki bilgileri iletmeyi içeren ilerlemeleri Rapor Etme"

Bu metodun proje yönetim süreci ise şu beş özelliği içerir.

(a) "Çekirdek metotların ve proje ekibinin belirlendiği Baştan Başa Bir Model Geliştirme,"
(b) "Özellik listesi çıkarma,"
(c) "Özelliklere göre planlama,"
(d) "Özelliklere göre tasarlama,"
(e) "Özelliklere göre gerçekleştirim."

Diğer metotlar gibi somut olarak işlevselliği bulunan yinelemeli süreçlerden oluşur [36, 37].

1.5.6. Dinamik sistem geliştirme

Ara ürüne dayalı bu metot iş sahası çalışmalarıyla başlar. Daha sonra fizibilite çalışmasıyla bu yöntemin uygunluğu kontrol edilir. Dinamik Sistem Geliştirme (DSG) metodunun diğer bölümleri ise yinelemeli ve eklemeli üç bölümden oluşur:

(a) “fonksiyonel model çevrimi; analiz dokümantasyonu ve fonksiyonel prototipleri içerir,”
(b) “tasarlama ve geliştirim çevrimi; tasarım prototiplerini içerir”
(c) “kodlama çevrimi, kodlamayı, son kullanıcı eğitimini ve kabul sürecini içerir.”

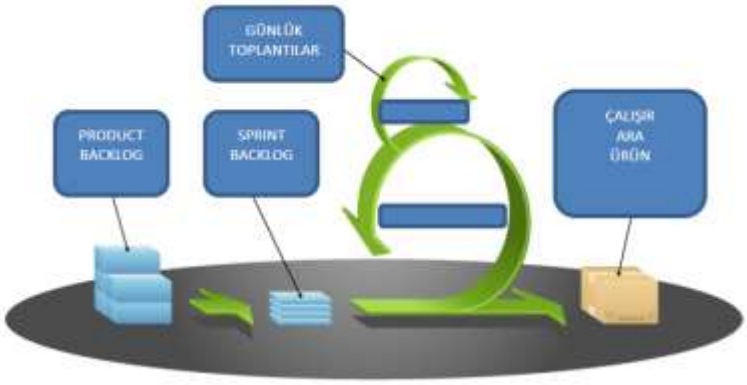
DSG geliştirici etkileşimini, ara ürünleri, sorumluluk alan bir ekibi, ihtiyaç temelli çalışmayı, iş birliğine dayanan ekip çalışmasını, eklemeler boyunca oluşan ara ürünü test yapmayı hedefler. Bu yinelemeler 2 ile 6 hafta arasında değişmektedir. Müşteri ile etkileşimin yüksek olduğu, geliştirici ekibin kesin olarak tanımlanabildiği, sürelerin net belirlendiği, ihtiyaçların önem derecesi doğrultusunda sıralandığı, ihtiyaçların belli olmadığı zamanlarda uygulanabilir [36]. Küçük gruplardan oluşan büyük projelerde kullanılabilir. Dinamik Sistem Geliştirme metodunun arkasındaki temel düşünce, üründeki işlevsellik miktarını sabitlemek, ardından sabitlenen işlevselliğe ulaşmak için zaman belirlemektir [36].

1.5.7. Scrum

Scrum yaklaşımı, sistem geliştirme sürecini yönetmek için geliştirilmiştir. Scrum, esneklik, uyum ve verimlilik düşüncelerini yeniden tanıtan bir yaklaşımla sistem geliştirme süreci için endüstriyel süreç kontrol teorisi düşüncelerini uygulayan ampirik bir yaklaşımdır [37]. Bu yöntem geliştirmeden çok geliştirmenin yönetim biçimine daha çok önem verir. Scrum kısa sürecek geliştirmelerde daha etkili olurken uzun sürecek geliştirmelerde ise geliştirmeyi koşular şeklinde bölerek yapmayı önerir ve her koşuda ne durumda olduğumuzu kontrol etmemizi ister. Bu kontroller geliştirmedeki durumumuzu bize aktarır ve takımdaki koordinasyonu arttırır. Projenin başında uygulanabileceği gibi ortasında ya da geliştirmenin sorunlu olduğu yerlerde kullanılabilir. Bu yöntem proje geliştirme dışında başka alanlarda da uygulanabilir [37].

Bu metot geliştirme başındaki işlere çok önem vermeden geliştirme aşamasında geliştirme ekibinin oluşabilecek farklılıklara daha iyi refleks gösterebilmesine daha çok önem verir. Scrum metodu koşulardan oluşur. Bu koşuların öncelikle süreleri belirlenir ve koşu sonlarında bir sonraki koşu da neler yapılacağı belirlenir. Ayrıca her gün geliştirmedeki durumumuzu aktardığımız günlük scrum toplantılarından oluşur. Koşu bitiminde oluşturulan geliştirme müşteriyle beraber kontrolü yapılarak bir

sonraki koşuya geçilir. Önemli nokta ise koşu'daki işler bitmeden sonraki koşu'ya geçilmez [38]. Scrum yönteminin döngüsü Şekil 1.5 de gösterilmiştir.



Şekil 1.4. SCRUM yöntemi

En fazla 15 kişiden oluşan ekipler geliştirmeyi yapmaktadırlar. Ekip geliştirmenin başında ihtiyaçların neler olduğunu yani geliştirmeyi tamamlamak için gerekli maddeleri belirler. Buradaki önemli nokta gereksinimlerin bayrak teslimi(koşu) içinde değişmez olduğudur. Sabahları ilk iş geliştirme ekibiyle günlük scrum toplantıları yapılır. Bu toplantıda işler ile işlerin gidişatı konuşularak oluşan farklılıklar karşılaşılan sorunlar tartışılır. Takımlar müşteriyi de katarak olabilecek en az belge ile direk olarak geliştirmeye konsantre olarak oluşacak ürünün bitirilmesini sağlar [38]. Böylece projenin olabilecek en yakın tarihte teslim edilmesinin önü açılmış olur [39].

Bu yöntem geliştirmelerin tahmininin zor olduğu yani sürekli farklılık gösterebilecek geliştirmelerde yapılmasında iyi sonuçlar vermektedir [40]. Bu metot diğer bu alandaki metotlara yakın olsa da en yakın olduğu metot EP'dir. Yani EP ile ortak olarak geliştirmelerde kullanılabilir.

Metodun özünde "Roller", "Toplantılar" ve "Kavramlar" bulunmaktadır.

Scrum'da bulunan Roller

(a) İş/Ürün Sahibi: Müşteri ile birebir iletişimde olan geliştirmede hangi özelliklerin olduğu ve bunların önem derecesinin neler olduğu yani kısacası geliştirmenin sınırlarına karar veren ürün sahibidir [41].

(b) Scrum Yöneticisi: Takımın bütün koordinasyonunu sağlayan metodun nasıl uygulanacağını belirleyen ve bunlara uyulmasına önem veren ayrıca geliştirmeden yükümlü olan takımın başındaki yöneticidir [41].
(c) Geliştirme Ekibi: Sürekli koordineli çalışan en fazla 15 elemanı bulunan ve her zaman geliştirmeyi tamamlamaya odaklanmış takımdır. Bu takımdaki her birey her koşu planlamasında kendi seçtiği işleri koşu sonuna kadar tamamlamaya uğraşan geliştirme yapan ekip üyeleridir [41].

Scrum'da gerekli Toplantılar

(a) Planlama Toplantısı: Takımın koşu başında koşu sonunda teslim edebileceği işleri belirlemek üzere yaptığı toplantılardır. Takımın sıradaki koşu için geliştirebileceği özellikleri önem derecesine göre aldıkları toplantılardır. Ürün Sahibiyle beraber koşu hedeflerine karar verirler [41].
(b) Günlük Toplantılar: Sabahları çok kısa süren bu görüşme takımın bir önceki günde neleri yaptığını kısaca anlattığı toplantılardır. Bu görüşmeler gidişatın ne durumda olduğundan haberdar olabilmek için yapılır. [41].
(c) Koşu Değerlendirme Toplantısı: Koşu bitiminde gerçekleştirilir. Takımın koşu sürecinde gerçekleştirdiği işlerin anlatıldığı görüşmelerdir. Takımın başındaki kişi toplantıda ilk önce koşu hedefini takıma söyler ve yapılan geliştirmeyi göstermeye çalışır. Görüşmede yapılan işler konuşulur ve başarılı mı başarısız mı olunduğu belirlenir. Koşu gereksinim setlerinde belirtilen değişiklikler bu görüşmelerde belirlenir [41].
d) Koşu Retrospektif Toplantısı: Bu görüşmeler ihtiyaç görüldüğü durumlarda gerçekleştirilir. Bu ihtiyaçlar koşu sonlarındaki toplantılarda bazı şeyler yolunda gitmediği göründüğü durumlarda gerçekleştirilir. Bu görüşmelerde takımın neleri iyi ve kötü yaptığı noktaları görüşülür. Bunlar bir sonraki koşularda düzeltilmeye çalışılır. Bir sonraki Retro da bu noktalarda ne gibi gelişmeler olduğu takip edilir [41].

Scrum'daki Kavramlar

Ürün Gereksinim Seti: Ürünün geliştirilmesi için gerek duygulan herkesin görebildiği maddelerden oluşan listedir. Burada ihtiyaçlar önem derecesine göre belirlenir ve listenin son hali oluşturulmuş olur [41].

Koşu Gereksinim Seti: Bir önceki bölümde yapılacak bütün işler belirlenmiştir. Bu bölümde ise belirtilen koşu sürecinde geliştiricilerin yapabileceği işler önem derecesine göre belirlenip alınır. Bu işlem sadece takım üyeleri tarafından yapılır. Dışarıdan hiç kimse olmaz. Her üye kendi aldığı işlere odaklanır ve koşu sürecinde bunları tamamlamaya uğraşır. Alınan işler koşu boyunca korunur yani işler alındıktan sonra koşu boyunca farklı işler alınamaz [41].

Koşu İlerleme Grafiği: Burada koşu boyunca alınan işlere süreler verilmiştir. Bu şekilde koşu boyunca alınan toplam süre bellidir. Her gün yapılan görüşmelerde takım üyesi o gün ne yapmışsa o işin süresinden düşerek koşu boyunca alınan işlerin ne kadarının yapıldığı bu grafik sayesinde takip edilebilir. Burada belirtilen süre gerçek anlamda süre değildir. Buradaki süreden kasıt işin tamamlanma yüzdesidir. Bu değerler üzerinden değişiklik yapılır [41].

2. OYUNLARDA KULLANILAN ÇEVİK YÖNTEM ÖRNEKLERİ

Bu bölümde çevik yöntem ile geliştirilen oyunların sonuçlarına yönelik araştırmalar yapılmaktadır.

Teknolojideki son gelişmeler giderek daha karmaşık ve gerçekçi oyunların geliştirilmesini sağlamaktadır [42].

Artan bu karmaşıklık geliştiricilerin daha fazla risk almalarına yol açmaktadır. Sonuç olarak, bir oyunun mümkün olan en az gecikme ve hatalarla geliştirilebilmesini sağlamak için teknik ve yöntemlerin kullanımında artan bir talep yaratır [43]. İhtiyaç duyulan teknikler ve yöntemler aynı zamanda oyunun başarısı için önemli olan "eğlenceyi" keşfetme sürecini de kolaylaştırmalıdır.

Her ne kadar bazı geliştiriciler yazılım geliştirmede zaten bilgi sahibi olsalar da oyun geliştirmenin başarısını engelleyebilecek daha farklı noktalar vardır. Bu durum, proje yönetiminde büyük sorunlara, halihazırda zaten yüksek olan maliyetlerin daha da artmasına ve gecikmelere yol açmaktadır. Oyun geliştirmeye odaklanmış bir metodolojinin kullanılması bu sorunların önlenmesine yardımcı olabilir. Bu şekilde, bazı yinelemeli yöntemlerin kullanımı bu sektörde artmaktadır [44] ve çevik metodolojiler popüler hale gelmektedir.

İncelediğimiz [44] makalesinde, oyun geliştirme için Scrum ve Ekstrem programlamaya dayanan çevik bir metodolojinin bir yaklaşımını sunmak ve mevcut iki çevik metodolojiyi gözden geçirmektedir.

Oyunda proje ekibinin "eğlence" faktörüne odaklanması ve aynı zamanda oyunu geliştirmeye odaklanması gerekmektedir. Bu iki faktöre aynı anda odaklanmak bir sorundur. Bu soruna basit bir çözüm olarak, çevik yöntemlerin dayandığı yinelemeli metodolojileri benimsemektedir. Bu metodolojiler, uygulanan özellikler hakkında erken bir geribildirim almayı sağlar, böylece gerektiğinde sorun olan noktalarda değişiklik yapmak daha kolaylaşır [44]. Diğer bir avantajı ise, oyun yaratmada rol oynayan herkes arasındaki iletişimi ve iş birliğini kolaylaştırması olmuştur [44].

2.1. Oyun Geliştirme Problemleri

Genel olarak yazılım geliştirmede bulunan sorunlar iyi bilinmektedir, ancak oyun endüstrisini etkileyen gerçek konular hakkında çok az şey bilinmektedir. Petrillo'ya göre [45] literatürde bildirilen ana problemler kapsam, planlama ve sıkışık iş süresi ile ilgilidir (aşırı iş yükü süreleri için kullanılan, genellikle son teslim tarihine yakın zamanlarda meydana gelen bir terim). Birtakım oyunlar ve çeşitli büyüklükteki projelerde yayın sonrası yapılan bir anket çalışmasından sonra, karşılaşılan problemlerin büyük çoğunluğunun proje yönetim problemleriyle ilgili olduğu keşfedildi [45].

Bu sorunların bir kısmı çoğu zaman multidisipliner ekiple ilişkilendirilir. Multidisipliner ekip, geliştirmeyi yapan kişilerin diğerlerinin yaptığı işle ilgilenmeden kendi işini yapmasıdır. Bu sebeple multidisiplinerlik, yaratıcılığı teşvik eden bir ortam oluşturmasına rağmen, “tasarımcılar” ve “geliştiriciler” arasında ayrım oluşturabilir ve tüm ekip arasında etkili bir iletişim kurmada zorlanılabilir [45]. Diğer bir nokta ise, oyunun eğlencesi gibi bazı unsurları belirlemede zorluk çekmektir; bu hedefe ne zaman ulaşılabileceğini açıklayan etkili bir yöntem yoktur [45].

Bu problemlerin çözümü için çevik yöntemlerin kullanılması, prototip oluşturma ve oyunun içeriğine bir iletişim hattı oluşturma gibi Kanode ve Haddad [46] tarafından önerilmektedir. Bu öneriler hakkında ayrıntılı bilgi için, lütfen Kanode ve Haddad’ın çalışmalarına bakınız [46]. Yaygın olarak karşılaşılan sorunların bir kısmı, aşağıda önerilen metodolojiler uygulanarak çözülebilir.

2.2. Ekstreme Oyun Geliştirme

Ekstreme Oyun Geliştirme [47] bir Ekstrem programlama uyarlamasıdır. Demachy'ye göre, odak noktası EP'yi oyun tasarımına ve çoklu ortam içeriğinin oluşturulmasına nasıl uyarlayacağımız ve oyunun belirli unsurlarını otomatik olarak nasıl test edeceğimizdir (“eğlence faktörü” gibi) [47]. Takımı bir bütün olarak ele alır ve oyun tasarımının öğelerini, tasarımcıları, programcıları ve hatta bu iki ekip arasındaki etkileşimi ve iletişimi kolaylaştırmak için kullanmaktadır [47].

2.3. Oyun-Scrum

Schwaber [48] 'e göre Scrum, proje yönetimine odaklanan bir çerçevedir. Bu yöntem kısaca her şeyi sorunsuz bir şekilde yapılabilmesi için görev dağılımını koordine eder ayrıca diğer çevik yöntemler ile uyumlu bir şekilde çalışabilmektedir [48]. Bu sebeple uygulamada EP ile Scrum ortak olarak kullanılmıştır. EP proje yönetiminden çok proje mühendisliğine odaklanmıştır. Proje yönetimini bu konuda deneyimi olan kişiler kontrol eder ve oyun geliştirmede az veya hiç tecrübesi olmayan insanların bu yöntemi içselleştirmesine odaklanır [48]. Proje yönetimi, oyun geliştirmenin önemli bir noktası olduğundan, ekibin Scrum'ın yinelemelerini ve yapılan toplantıların neden önemli olduğunu anlamalarını hedef almışlardır. Kısaca projenin başarısı için önemli olan noktalar vurgulanmaktadır [48].

Çevik metodolojinin gelişim aşamaları boyunca, tüm yinelemeler yani koşular, yazılım geliştirmek için temel adımların yaklaşık olarak aynı dağılımına sahiptir. Ancak Keith'e göre [49] oyun geliştirmedeki işlerin dağılımı, yinelemeler sırasında eşit olarak dağıtılmamıştır. Bu fikri desteklemek için, Oyun Scrum, aşağıdaki gibi üç aşamaya ayrılmıştır.

2.3.1. Üretim öncesi

Bu aşamada, oyun “keşfedilmelidir”, yani oyun hedefinin gerçekte ne olduğu ve içindeki “eğlenceli faktörü” ne olmalıdır [49]. Burada oyun kavramı, programlama dili, platform ve diğer tanımlara karar verilir. Kanode ve Haddad’ın belirttiği gibi [49], “iyi bir ön prodüksiyon, prodüksiyon aşamasında “eğlence” unsurunu bulma ihtiyacını azaltır ve ekibin oyunu test etmekten ziyade oyunu geliştirmeye odaklanmasına izin verir.”.

Bu aşama, yaratıcılığı fazla engellemeden üretim aşamasının alacağı yönü tanımlar. Burada yapılacak iş, genellikle deneme yanılma yoluyla ideal oyun konseptini ve tasarımını bulmak olacaktır. Bunun için fikir geliştirmek ve yeni fikirleri bir araya getirmek için beyin fırtınası yapılması önerilir. Prototip geliştirmek, aynı zamanda oyunun veya bir kısmının sunabileceği “eğlenceli faktörünün” ön izlemesine yardımcı olacaktır. Prototip, kullanıcıların oyunu test etmesi için gereken özellikler için basit bir ön gösterim sağlamalıdır [49]. Genellikle, burada üretilen kod tekrar kullanılmaz

ve prototip atılır. Bu anlatılanlar ışığında incelediğimiz örnek uygulamada yapılanlar aşağıdaki gibidir.

İlk beyin fırtınasında, geliştiricilerin her biri için belirli bir bölümü veya yazılım mühendisliği kavramını ele alan, farklı yaşlardaki çeşitli mini oyunlar sunma fikri tartışıldı. Oyun bu aşamada taş çağında geçen bir oyun olması planlanmıştı [49].

Daha sonra, beyin fırtınası sırasında tartışılan fikirleri test etmek için prototip yapıldı. Prototipte, oyunun senaryosu, bazı ekran geçişlerini ve oyuncu gereksinimlerinin sunumunu temsil eden bir çizim yapıldı. Başlangıçta iyi bir fikir gibi görünse de taş çağında geçen bir yazılım sisteminin benzer gereksinimlerine sahip olan görevlerin işlenmesinin çok zor olduğu keşfedildi, bu nedenle beyin fırtınası sonucunda oyunun orta çağda geçen bir oyun olmasına karar verildi [49].

Nispeten kısa bir süre içinde geliştirilecek olan basit bir oyun olduğundan, bir oyun tasarımı belgesi yazılmamıştır. Dolayısıyla, mini oyun fikri ekip içinde çok açık olduğu için, bu mini oyunun gereklilikleri doğrudan bir ürün gereksinim listesi olarak tanımlandı.

2.3.2. Oyun tasarımı dokümanı

Oyun tasarım belgesinin oluşturulması, oyunun yapım öncesi aşamasında gerçekleştirilmesi gereken, projenin kapsamını (ve oyunun tüm gelişimini ve testini) yönlendirmekten sorumlu olmak için atılmış önemli bir adımdır. Zayıf bir oyun tasarım dokümanı özellik değişmelerine neden olabilir ve bunun sonucunda gecikmelere ve ek maliyete sebep olabilir.

Bir oyun tasarım dokümanı oluşturmanın standart bir yolu olmamasına rağmen, Schuytema [49], bu dokümanın oyunun tüm yönlerini kapsaması gerektiğini, böylece geliştirme ekibinin tasarımlara başlayabileceğini belirtir. Oyundaki eşya, nesne ve karakterleri tanımlarken, sadece yaptıkları işleri değil, aynı zamanda neyi etkilediklerini, nasıl etkileşimde olduklarını ve oyundaki rol ve davranışlarını da belgelendirmelidir. Belge genel olarak oyunun son hali gibi düşünülse de belge sürekli değişebilir. Bununla birlikte, kişi değişikliklerin risklerini değerlendirmeli ve teslim tarihinin hala karşılanıp karşılanamayacağını değerlendirmelidir.

Bu belge daha sonra üretim aşamasında bir ürün gereksinim listesine çevrileceğinden, küçük oyunlar için isteğe bağlı olarak gereksinimler doğrudan ürün gereksinim listesi olarak çevrilebilir. Bu kısım kısa tutulursa, üretime daha hızlı geçileceğinden ekibe zaman kazandırabilir ancak özellik değişmesi riskini de artırabilir veya çok eğlenceli bir oyun olmayabilir.

2.3.3. Üretim

Bu aşamada, zaten iyi tanımlanmış bir proje kapsamı olmalı ve dolayısıyla oyunun gerçekte ne olduğu ve gerçekte ne yapılması gerektiği hakkında iyi bir fikir olmalıdır [43]. Burada oyun tasarım dokümanı bir ürün gereksinim listesine çevrilmelidir. Her yinelemede, kalan en önemli maddeler yineleme süresine göre yüksek öncelikten düşük önceliğe göre geliştiricilerin üstüne atanmalıdır [49].

Birçok geliştiriciden oluşan ekipler için daha iyi bir yaklaşım düşünmek gerekiyor. Sıklıkla geliştirilecek bir görev genellikle birkaç geliştirici tarafından gerçekleştirilir. Örneğin, tasarımcı çalışmalarını animatöre teslim eder, animasyonda bittikten sonra görevi ses geliştiricisine iletir vb. Keith [49]. Geliştirilen oyundan sorumlu olanlar Kanban'ın kullanılmasını önermektedir. Bu koşunun sonunda, tüm çalışmaların bitmesini gerektiren Scrum'ın aksine, halen geliştirilmekte olan bir işin olmasına izin verir [49]. Amaç içerik oluşturmada sürekli bir akış sağlamaktır.

Ön üretimin sonunda, “EngReq”, orta çağlardaki köyde geçen mini bir oyundur. Oyunu, geliştiren ekipteki kişilerin deneyimi yoktur veya çok azdır. Bu nedenle, sahne ve karakterlerin tasarımı dış kaynaklardan sağlanmıştır, ancak yine de tasarımcının geliştiricilerle düzenli olarak buluştuğu ve her iki tarafın da fikirlerini sunarken olabildiğince açık olmaya çalıştığı yinelemeli model uygulanmıştır.

2.3.4. Post prodüksiyon

Oyun tamamlandıktan sonra, yapılan testler oyunun kalitesini ve eğlenceliğini anlamaya yardımcı olur. Fakat burada postmortem denilen, yayın sonrası kullanıcıların geribildirimlerine odaklanacağız. Postmortemler önemlidir, çünkü kullanılan geliştirme sürecinin güçlü yanlarını, zayıf yanlarını, ortaya çıkan sorunları ve iyileştirme önerileri hakkında bilgi sahibi olmanın en önemli yoludur. Bu

geribildirimler sayesinde, gelecekteki projeler için daha iyi bir tahmin yapılabilir ve süreç için gerekli düzeltmeler yapılabilir [49].

Myllyaho [50] postmortem analizinin yararlarını, avantajlarını ve Gamasutra'nun web sitesinde bulunan postmortemlerden bahseder. Çalışmalarında tanımlandığı gibi, genellikle postmortemler, proje yönetimi alanları dahil olmak üzere çözümler, iyileştirmeler, uygun araçlar vb. içeren bir listeyi anlatmaktadır [50].

Ayrıca, postmortemler deneyimlere ek olarak aşağıdaki oyun projesi ölçümlerini de içerir. Bunlar yayıncı, geliştirici, tam zamanlı geliştiriciler, yarı zamanlı geliştiriciler, yüklenicilerin sayısı, geliştirmenin süresi, çıkış tarihi, oyun platformları, geliştirmede hangi donanım ve yazılımı kullanılmıştır bütün bunları içerir [50].

Bu anlatılan yönteme göre örnek uygulama aşağıda verilmiştir.

Örnekleme amacıyla, bu bölümde Game-Scrum, yazılım mühendisliği alanındaki birkaç mini oyundan oluşan bir eğitim oyunu olan EngGame'in geliştirmesi uygulanır.

Oyun yayınlandıktan sonra belirli kişiler tarafından test edildikten sonra hazırlanan bir anket ile mini oyunun güçlü ve zayıf yönleri ve iyileştirme önerileri hakkında fikirler alınmıştır. Projenin geribildirimi ile birlikte bir EngReq postmortem üretildi.

2.3.5. Sonuç

Artan maliyetler ve gereksinimlerin yüksek değişkenliği ile oyun geliştirme için yapılan, yatırımın bize geri dönüp dönmeyeceğini en kısa sürede bilmek önemlidir. Oyunda neyin "eğlenceli" geldiğini ve iyi çalıştığını bilmek, başarının ön şartıdır. Ancak, geleneksel geliştirme yönteminde bu durumun anlaşılması, yalnızca geliştirmenin son aşamasında, yani çok fazla zaman ve para harcandığında ve değişiklik yapmanın büyük riskleri olduğunda ortaya çıkar. Yinelemeli bir metodoloji, özelliklerin en kısa sürede hazır olmasını sağlar ve böylece oyunun "eğlencesini" keşfederek çalışmak daha kolaydır, sonuç olarak çevik yöntemler gerçekte neyin önemli olduğuna odaklanmanıza izin verir.

Kolayca ölçülemeyen “eğlenceli faktörü” bulmaya ihtiyaç duyulan bir oyun geliştirme ortamının ihtiyaçlarına odaklanan çevik metodolojiler için çok az araştırma vardır. Game-Scrum tarafından yapılan yaklaşımlardan bazıları şöyle özetlenebilir:

• Proje kapsamı: Beyin fırtınası, prototipleme ve oyun tasarım belgesini ön prodüksiyonun bir parçası olarak kullanmak, oyunun “eğlencesinin” keşfedilmesini kolaylaştırır ve yinelemeler bu faktörün gelişmesini sağlar. Ayrıca, bu teknikler oyunda özellik değişmesi oranını azaltabilir.

• Proje yönetimi: Oyun geliştirmek için Scrum çerçevesini kullanmak, Oyun-Scrum'ın temelini oluşturur. Scrum'da olduğu gibi, projenin geribildirimlerinin sürekli değerlendirilmesi oyunu iyileştirmeye yardımcı olur.

• Takım organizasyonu: Takımın güçlü ve zayıf yönlerine göre seçebileceği organizasyonu geliştirmek için bazı alternatifler önermek.

3. YÖNTEM

Önceki bölümler de çevik yöntemlerin tarihçesinden ve bugüne kadar geliş sürecinden bahsedilmektedir. Ayrıca örnek bir oyunun çevik yöntem ile incelenmesi açıklanmıştır.

Bu bölümde çevik yöntem kullanılarak iki tane oyun yapılacaktır. Çevik yöntem metodu olarak da scrum yöntemi seçilmiştir. Scrum yönteminin incelememizin sebebi ise en çok kullanılan ve içeriği nedeniyle geliştirmelere en uygun olan proje yönetim biçimi olmasıdır. Geliştirmenin en iyi şekilde sonuçlanması için ihtiyaçları belirlemek adına takım ile müşteri arasındaki iletişimin en iyi düzeyde olduğu metotların başında geliyor. Aşağıda öncelikle scrum yönteminin tarihsel gelişim süreci ve nasıl uygulanacağı detaylı bir bicinde anlatılmıştır.

3.1. Tarihçe

Hirotaka Takeuchi ve Ikujiro Nonaka 1986 yılında, proje üretiminde aşamaların uyum içinde olduğu ekibin beraber konsantre olarak en verimli şekilde geliştirme yapılabilecek metotları ortaya koydular. Çevik metotlar birden çok alanda kullanılması amacıyla ortaya çıkmış asıl hedefi değişime ve farklılıklara olabildiğince hızlı bir şekilde yanıt verebilmektir. İlk olarak adı rugby ve holistik olsa da her şey takımın birbirini kontrol eden ve geliştirmeyi bitirmeye odaklanması şeklinde meydana gelmiştir [51].

İlk olarak “Ken Schwaber” 1990’larda bu metodu “Easel” adlı şirketinde kullandı ve ilk olarak Scrum ismi burada ortaya çıktı [51]. Daha sonra “Schwaber” ve “Sutherland” 1995 senesinde yaptıkları geliştirmede Scrum yöntemini aktardıkları bir makaleyi ele aldılar ve bu da insanlar tarafından scrumın içeriğini öğrenmeye dahil olan ilk yazı niteliğindeydi [52]. Bu yöntem hemen hemen her geliştirmede başvurulan iyi sonuçlar veren en önemli yöntemlerden biri olmuştur.

Bu yöntem firmaların birbiriyle olan yarışları sebebiyle yaptıkları geliştirmelerde daha iyi sonuçlara ulaşmak amacıyla çok sık olarak başvurulan yöntem olmaktadır. [53]. Bu

konuda geliştirme işleri ile uğraşan analistler bu yöntemin diğer yöntemlere oranla daha başarılı sonuçlar elde ettiğini kabul etmektedirler [53].

Bu yöntem öncelikli olarak önemli olan noktaları bitirmeyi hedefler. Sürekli geribildirim sağlayarak oluşabilecek hataları veya değiştirilmesi istenen noktaların erken farkına varılmasını sağlar.

Yani buradaki geribildirimler oluşabilecek değişikliklerin zamanında öngörülmesini ve oluşabilecek sıkıntıların önceden görülmesini sağlar. Bu yüzden scrum değişen şartlara karşı dayanıklı ve oldukça başarılıdır [53].

Scrum oluşabilecek sıkıntıları öngörmek ve önlemek için tasarlanmış olup, yinelemeli süreçlerde geçmişte oluşan durumları göz önüne alarak ilerlemeyi hedefler. Bu ilerlemeyi sağlayan hususlar aşağıda belirtilmiştir [53].

3 temel noktadan oluşur:

a) Şeffaflık: Geliştirme yaparken oluşabilecek sorunları ve bu sorunları aşmamızda bize olanak sağlar. Yapılan işlerde sadece şeffaf olması yeterli değil aynı zamanda ne yapıldığının da iyi bilinmesi gerekir. Böylece geliştiricilerin yaptığı işler ile koşu da verdiğimiz işlerin ne derece orantılı gittiğinin gözlemlenmesi konusunda net bir sonuç gözlemlenebilir [53].
b) Denetleme: Bu nokta önemlidir çünkü geliştirme sık denetlenirse süreçte ortaya çıkabilecek sorunlar en aza indirgenmiş olur. Denetlemenin aralığı geliştirilen işin ne kadar farklılığa uğradığına göre belirlenir. Burada oluşabilecek farklılıklar sonucu denetlemenin gereğinden fazla olması durumu ortaya çıkabilir. Ancak bu durum çok az rastlanılan bir durumdur. Bu durumu etkileyen en önemli faktör ise geliştiren kişilerin işinde ne derece uzman olduğuyla ilgilidir [53].
c) Adaptasyon: Bu konuda eğer yaptığımız iş istenilen işin isteklerini karşılayamama gibi bir sonuç doğuruyorsa geliştirmenin başarısızlıkla sonuçlanacağı öngörülebilir. Bu gibi durumla karşılaştığımız zaman denetleme mekanizması devreye girer ve geliştirmeyi doğru bir şekilde tamamlamamız konusunda bize destek vermelidir. Yani adaptasyonu sağlayan etmen burada denetleme olduğu gözlemlenmektedir [53].

Bu yöntemde geliştirme planlaması aşağıda belirtilen noktalardan oluşur [53];

• Müşteri gereksinimleri: Geliştirilen projenin hangi özellikleri içereceğinin belirlenmesidir.
• Zaman baskısı: Geliştirmedeki zamanı daha aza indirgemek için yapılabilecek noktaların planlanmasıdır.
• Mücadele: Benzer geliştirmelerle yarışabilmek için gerekli olan ihtiyaçları belirlemek ve bunların getiri ve götürülerinin hesaplanmasıdır.
• Kalite: Ortaya iyi bir ürün koyabilmek için gereksinimlerin belirlenmesidir.
• Vizyon: İleride çıkabilecek sorunları belirlemek adına vizyon gözden geçirilir ve sorun çıkabilecek noktalar öngörülmeye çalışılır.
• Kaynak: Geliştirmeyi tamamlayabilecek ihtiyaçların belirlenmesidir.

Yukarıdaki noktalar projenin en başında belirlenir. Sonradan farklılık gösterebilir. Bir yöntemin başarılı olabilmesi için yukarıda belirtilen noktalarda sorun çıktığı zaman hızlı bir şekilde giderebilmesi ve çıkan sorunları iyi bir şekilde yönetebilmesi gerekmektedir. [53].

3.2. Kendi Kendini Organize Etme

Bu husus aslında bir ekip halinde dışarıdan bağımsız olarak çalışabilmeyi ifade eder. Kişilerin bağımsız olduğunu ifade etmez. Yani kısaca ekip kendi içinde iletişim halindedir ama dışarıdan bağımsız bir şekilde çalışabilmelidir bunu ifade eder [54]. Bu yüzden ekip geliştirmeyi teslim edebilmesi adına kendi içinde bağımsızlık verilir. Rahat çalışabilmeleri için bu durum gereklidir. Bu yüzden ekip bir problemle karşılaştığı zaman kendi içinde bu durumu halletmesi için özgürlük tanınır. Yani kısacası takımın sorunlarla baş edebilmesi beklenir [54].

Bu yöntemde ekiplere verilen sorumluluktan ötürü ekibin özgüveni ve iş bitiriciliği üst düzeyde olması beklenir [54].

Bu sebeple ekipler ilk başta geliştirmeye başlamadan önce bu sorumlulukları alacakları için bir motivasyon olması adına bazı beklentileri olacaktır. Bu yüzden hem geliştiren ekip hem de iş sahibi eski tutumlarında değişikliğe gitmeleri gerekmektedir [54].

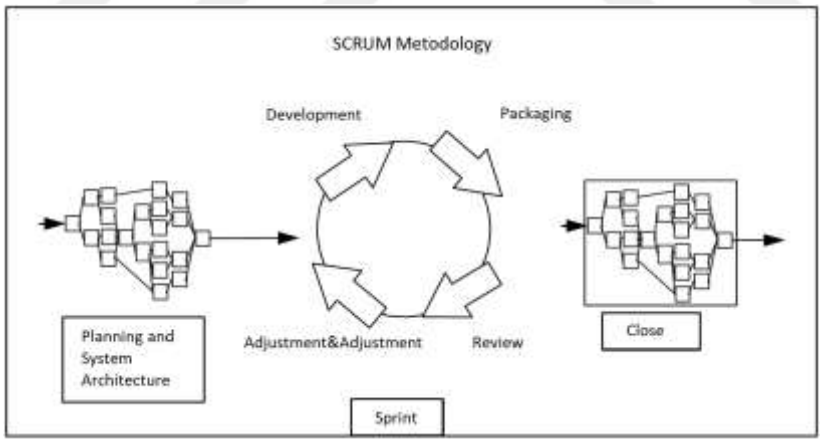
Bu husus bir istek değil bir zorunluluktur. Eğer bu hususu ekipler karşılamazsa bu metodun başarılı olma ihtimali oldukça düşüktür. [54].

3.3. Scrum'ın Aşamaları

Scrum evreleri ilk önce geliştirmede ihtiyaç duyulan ekipmanları belirlemek ile başlar daha sonra müşteri ile genel hatlar üzerinde anlaşılır ve sonrada geliştirme evresinin yapılmasıyla devam edilir. Bu evrelerden ilk olarak hazırlık evresi yapılır. Daha sonra geliştirme evresinde yinelemeli olarak ilerlenir. Her yinelemede bir ara ürün oluşturarak gerekli kontroller yapılıp artımlı bir şekilde sonraki yinelemelere geçilir. En sonda bitiş evresi yer alır.

Bu yöntem geliştirmelerin başladığı evre ve sonuç evresi ortasında yinelemeli geliştirme evresinin bulunduğu bir yöntemdir. Bu evreler Şekil 3.1 de gösterilmiştir.

Planlama yani başlangıç ile kapatma yani bitiş evreleri ile yinelemeli evreler arasında düz bir şekilde devam eden yönetim mevcuttur [54]. Burada tüm evrelerde hangi işlemler yapılacağı belirli olmak zorundadır.



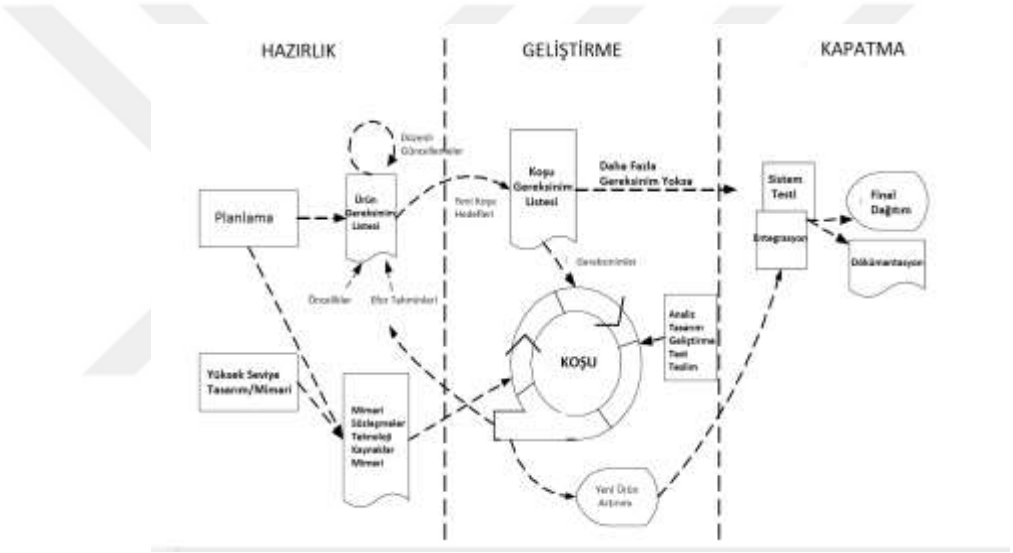
Şekil 3.1. Scrum aşamaları

Burada belirlenen her koşu planlıdır ancak koşu kendi içinde bağımsız bir yapıdadır. Bunun için her koşu sonrası bir test veya bir sorgulama işleminin olması elzemdir. Bu koşu sonlarında yapılan oluşabilecek sorunları önlemeyi ve en başarılı şekilde ürün oluşturmamızı sağlamayı hedefler [54]. Burada koşular her ne kadar kendi içinde bağımsız olsa da koşular değişebilir. Çünkü koşuda bir sorunla karşılaşılırsa bu durum bir sonraki koşuda giderilmek üzere takvime eklenebilir [54].

Oluşturulan her ara ürün koşular ilerledikçe değişiklik gösterebilir [54]. Yani ürün geliştirme durumuna göre değişiklik gösterebilmektedir. [54]. Kısaca scrum işin yapımını göstermez. İşin en verimli şekilde yapılabilmesi için takımlara yetki verir [55].

Her koşu sonrası oluşturulan ara üründe belirlenen değişiklikler sonraki koşularda önem derecesine göre tekrar planlanır. Bu şekilde ilerleyen koşularda belirlenen işler değil oluşan sorunlar ve beklemede olan işler önem derecesine göre alınarak ürünün oluşturulması sağlanır [55].

Scrum aşamasının detaylı şeması aşağıda Şekil 3.2'de gösterilmiştir;



Şekil 3.2. Scrum'da süreçlerin genel görünümü

Aşağıdaki bölümde bu aşamaların projemizde nasıl uygulandığını açıklayacağız.

3.3.1. Hazırlık aşaması

a) Planlama**:** Bu aşamada yapılacak geliştirmenin ihtiyaçları belirlenerek liste yapıldığı bölümdür. Bu aşamada daha önce yapılmamış yeni bir geliştirme ile karşı karşıya kalınırsa takımın geliştirmeyi yorumlaması gerekmektedir. Eğer böyle bir konu yoksa standart planlama yapılarak geliştirilmeye geçilebilir [55].

Aşamaları aşağıdaki gibidir.

• Gereksinim listesi çıkarılması

• Her koşu sonunda yapılacak dağıtımların tarihi ve fonksiyonel özelliklerinin belirlenmesidir.
• Risk değerlendirmesi ve risk kontrollerinin hazırlanmasıdır.
• Geliştirilecek projenin ekipmanlarının hazırlanmasıdır.
• Yapılacak geliştirme ile ilgili ileriki safhalarda bulunan adımlar için bir maliyet planı hazırlanmalıdır.

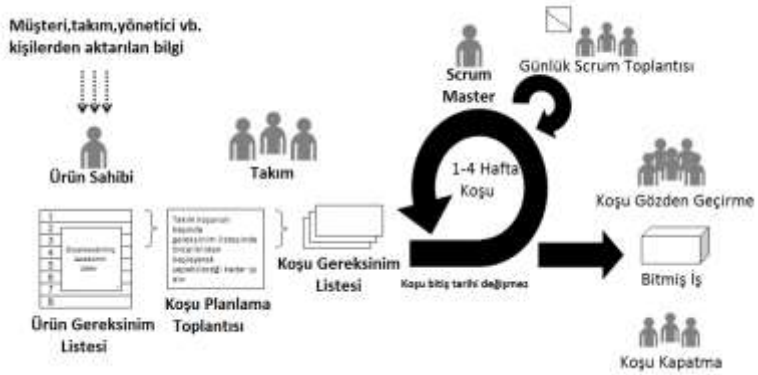
b) Mimari ve Yüksek Seviye Tasarım: Burada karar verilen senaryolardaki işlevlerin ne şekilde oluşturulacağına karar verilir ve yapılır. Bunun için işin uzmanı kişiler olması gerekmektedir [55].

Aşamaları aşağıdaki gibidir.

• İhtiyaç listesinin son haline göre yapılmalıdır.
• Yapılacak işler ağırlık derecesine göre sıralanmalıdır.
• Bu tasarımları yaparken ortaya çıkabilecek noktalar için liste hazırlanmalıdır.
• Yapılacak tasarımların istenilen her türlü özellikleri yerine getirecek şekilde tasarlanması gerekmektedir.

3.3.2. Geliştirme

Bu evre yinelemeli aşamalardan oluşur ve Şekil 3.3 de gösterilmektedir. Bu aşamada yönetimde bulunan kişiler yapılan işlerin takibi ile ilgilenirler yani yapılan işlere karışmazlar çünkü bu takımların sorumluluğundadır. Sadece takip ettikleri sırada bir problem gördüklerinde ekipleri bilgilendirirler [55].



Şekil 3.3. Scrum metodolojisi geliştirme evresi [55]

3.3.3. Dağıtım planlama toplantısı

Bu aşamada takımlar en az sorunla karşılaşarak projeyi nasıl tamamlayabileceğini kararlaştırır. Aşağıdaki evrelere karar verirler [55];

• En verimli ve en kaliteli şekilde projeyi nasıl tamamlayabiliriz?
• Ürün sahibinin geliştirmede istediği işlevleri ve hazırda bulunan işlevlerini en iyi şekilde nasıl yapabiliriz?
• Geri bildirimler ve iyileştirmelerle ürün sahibini ne kadar mutlu edebiliriz?

Yapılan toplantılarla varmak istediğimiz sonucu listedeki işlevlerin önem derecesini, oluşabilecek sorunları ve hali hazırda bulunan işlevlerin karar verildiği bölümdür. Tabi ki bu aşama koşu aralarında oluşabilecek sorunlardan dolayı sonraki koşularda önceliklere göre değişim gösterebilirler [55].

Bu aşama isteğe bağlıdır. Eğer bu aşama yapılmadan uygulamaya başlanılırsa her koşuda ürün gereksinimleri gözden geçirilmelidir. Her koşu önemli maddelerden başlanılarak artırımlı şekilde ilerlenir. Burada geliştirme esnasında ürün oluşturabilecek konuma gelindiğinde geribildirimler için sürekli ürün çıkarılmaya başlanır [55]. Bu yöntemde bu aşama genel olarak geleneksel metotlara oranla daha çok zaman alabilmektedir [55].

3.3.4. Geliştirme koşuları

Bu kısım geliştirme bitene kadar yinelemeli ve artımsal olarak birbirini tekrar eden koşulardan oluşur [55].

Bu evrenin önemli noktaları aşağıdaki gibidir;

• Koşu başında alınacak işlerin önem derecesine göre ekip ile kararlaştırılarak geliştirici üzerine alınması
• Bulunulan aşamada geliştirmenin durumu ile ilgili bilgilendirme veya yeterince ilerlenmişse bir ara ürün oluşturma
• Son olarak da ara ürün çıkaracak seviyeye ulaşana dek yinelemeli ve artımsal olarak koşulara devam edilmesi

Her koşu bir veya iki hafta ile tanımlanmış (bu süre proje büyüklüğüne göre değişebilir) özelliklerden oluşur. Koşu süresi ve alınan işler seçilen metotlara göre farklılık gösterebilir [55].

Koşu sırasında koşu ile ilgili herhangi bir madde özellik değiştirilmemelidir. Koşuda ilk önce hangi işlerin alınacağına karar verilen planlama ile başlayıp daha sonra ürünün özelliklerini kodlama ile devam ederek koşu sonunda ne kadar başarılı olduğuna ve ne kadar ilerlenebildiğine karar verilen toplantı ile son bulur. Koşular arası hiç zaman aralığı bulunmamaktadır. Koşular, koşu süresi tamamlanmadan iptal edilebilir. Bunu yapmaya sadece ürün sahibinin yetkisi vardır.

Her projede projenin iyi olup olmadığının anlaşılması için gerekli bir süre vardır. Bu süre çok uzun olmamalıdır. Eğer çok uzun olursa alınan riskler daha büyük olur. Koşular koşu planlama toplantısı ile başlar her gün günlük scrum toplantısı yapılır ve koşu bitirme toplantısı ile sonlandırılır.

a) Koşu planlama toplantısı

Koşu başlamadan önce yapılır [55]. Bu toplantıda ihtiyaç listesindeki özellikler kontrol edilir ve bu özellikler önem derecesine göre bu koşuda yapılacak işler arasına alınırlar. Bu işlemler ürün sahibi ile aynı paralelde yürümesi gerekmektedir. Bu şekilde olmazsa ileride oluşturulan ürünlerde tutarsızlık meydana gelir ve sorunlar daha büyük olur [55].

Takımlar koşularda yapılacak işleri aldıktan sonra aldıkları işlere bir süre vererek geliştirmelerin ne kadar süreceği konusunda bilgi verirler. Burada ekip aldıkları işi bitirmek zorunda değildir. Eğer aldıkları iş büyük çaplı ise bir kısmının biteceğine dair bilgi verir ve koşu içinde aldıkları işleri bitirmeye çalışırlar [55].

b) Günlük Scrum Toplantısı

Burada toplantı her gün kararlaştırılan zamanda çok kısa olarak gerçekleştirilir ve bütün ekip üyeleri dahil olmak zorundadırlar [55]. Ekibin geliştirme yaparken ne durumda olduklarını görmelerini sağlar. Her takım elemanı aşağıdaki noktalar hakkında bilgi verirler [55];

• “Dün ne yaptım?”

•” Bugün ne yapacağım?”

• “Önümde olan engeller ve karşılaştığım sorunlar neler?”

Bu toplantıda yönetici sadece koşu içinde ne durumda olduğumuz konusunda bilgi tutar. Eğer bir sorun varsa toplantı sırasında değil sonrasında destekte bulunmalıdır [55]. Bu görüşme bir tartışma havasında olmamalıdır. Sadece kişiler çıkıp yukarıda belirtilen 3 soruya cevap vermelidirler. Eğer bir anlaşmazlık olursa bu durumu görüşmeden sonra başka bir toplantıda konuşulup ortak bir noktaya varılmalıdır [55]. Bu toplantı bittikten sonra kişiler aldıkları işlerdeki sürelerinde gerekli azaltmaya gitmeleri gerekmektedirler. İşleyiş söylenildiği gibi yapılırsa koşu sonunda koşu da takımın işleri nasıl yaptığı azalan zaman grafiğinde gözlemlenebilecektir [55].

c) Koşu Kapatma Toplantısı

Bu aşama en önemli aşamadır. Bunun nedeni takımın koşuda ilerleme kat edip etmediği koşu sonlarında yapılan değerlendirmelerde ortaya çıkar. Bu kısım takım içindeki üyelerin birbirinden haberdar olmalarını sağlar ve projedeki ilerleyişin durumu hakkında herkesin net bir sonuca varabileceği yerdir. Ayrıca anlaşmazlık durumunda müsait bir tartışma ortamının oluşabileceği en önemli toplantıdır. Bu aşamanın önemi aslında şuradan daha net anlaşılır. Scrum sürekli yinelemeli ve artımlı bir metot olduğu için bu bölüm bu işlevin meydana gelmesini sağlayan kritik bir noktadır.

Proje bitene kadar bu aşamalar tekrar edilir. Proje bittikten sonra ürünü yaygınlaşabilmesi adına geliştirme hakkında bir kılavuz diyebileceğimiz belge oluşturulur. Test edilebilmesi için ilgili kişilere ürün teslim edilir.

4. HUYSUZ TOP

Yapacağımız oyunumuzda proje yönetimi olarak Scrum metodolojisi kullanılacaktır. Çevik yöntemin avantajlarından biri sürecin esnek olmasıdır. Fakat bu esneklik sürekli uygulama da var olan özelliklerin değiştirilmesine ve hatta yeni özellikler eklenmesine sebep olmaktadır. Bu nedenle uygulamanın bir bitiş tarihi belirlenmesi pek de mümkün değildir. Biz bu uygulamamızda çevik yöntem kullanılarak bitiş tarihinin tespiti için çalışma yapmaktayız. Bunun için planlama aşamasında yapılan süre tahminleri ile bitiş zamanındaki çıkan süreleri maddelere göre ayırarak çıkarım yapmaya çalışacağız. Aşağıda yaptığımız uygulamanın aşamalarını açıklıyoruz. Daha sonra çıkan verilere göre sonuçları inceleyeceğiz.

4.1. Üretim Öncesi

Bu aşamada oyunun temel özellikleri belirlenmiştir. Bunlar oyunumuz halkalar ve bir toptan oluşmaktadır. Her seferinde farklı renkte 3 tane halka gelecektir. Topun rengi hangisiyle aynı ise kullanıcı topu o halkanın içinden geçirmeye çalışılacaktır. Oyun sabit hızda ve toplam 100 tane halkanın içinden geçince sonlanacaktır. Oyunumuzun temel ilkeleri bu şekilde belirlenmiştir.

Oyunumuz Unity platformu kullanılarak C# dilinde kodlanacaktır. Masaüstü, Android ve iOS platformlarında oynanabilecek şekilde yapılacaktır. Prototip üretimi yapılmamıştır. Direk oyunun yapımı aşamasına geçilmiştir. Çevik yöntemin başarısının daha iyi test edilebilmesi için oyunda gerekli görünen yerdeki değişiklikler fark edildikten sonra planlanarak değiştirilecektir.

Geliştirme aşamasında oyunumuz test edildikçe bazı değişiklikler yapılmıştır. Bunlar ilk önce oyuna başlarken 3 tane can hakkı verilmiştir. Ancak 100 halka bu 3 tane can ile geçilmesi kolay olmamıştır. Bu yüzden her 10 halkada bir halkanın içinde yıldız sembolü bulunmaktadır. Oyuncu bunu alırsa 1 can daha eklenecektir.

Oyun oynandıkça sabit oyun hızı oyunu monotonlaştırdığı gerekçesiyle oyun giderek hızlanmaktadır.

Daha sonra oyunun tek bölüm olması oyunun oynanma süresini kısaltmaktadır. Bu nedenle oyunda bölümlendirmeye gidilmiştir. Toplam da 4 tane bölüm vardır.

Bölüm 1: Topun rengi sabit halka renkleri her seferinde rastgele olacak şekilde gelmektedir.

Bölüm 2: Topun rengi değişken halka renkleri her seferinde rastgele olacak şekilde gelmektedir.

Bölüm 3: Topun rengi sabit halka renkleri her seferinde rastgele olacak şekilde gelmektedir. Farklı olarak halkalar hareketli olarak gelmektedir. Yani halkalar durağan değildir.

Bölüm 4: Topun rengi değişken halka renkleri her seferinde rastgele olacak şekilde gelmektedir. Halkalar hareketli olarak gelmektedir. Yani halkalar durağan değildir.

En son olarak topun oyun halkalarından geçip geçmediği daha net anlaşılması için halkalar 3 boyutlu yapılmıştır.

4.1.1. Oyun tasarım dokümantasyonu

Bu oldukça önemli bir konudur. Çünkü tasarımda bir şey atlanırsa veya ihmal edilirse sonradan oyunumuzdaki bazı özelliklerin kaybolmasına neden olabilir. Bu durum oyunun en önemli özelliği olan “eğlence” faktörünün azalmasına sebep olur.

4.1.2. Ürün gereksinim listesi

Gökyüzü: Arka planda gökyüzü görseli planlanmıştır.

Bulut: Arka plan ile iç içedir. Ekranda rasgele gelecek şekilde bulut görselleri yapılmalıdır. Bunlar oyunun hızına bağlı olarak ekranda hareket edecektir.

Oyunun durumu: Ekranda kaç tane halka geçildiği ve toplamda kaç halka olduğunu kullanıcının görebilmesi için ekranın sol üst köşesinde bu görsel olacaktır.

Can: Oyunda kaç canımızın kaldığını gösterir. Ekranın sağ üst köşesinde olacaktır.

Oyun Oynanma Koşulları: Oyunun nasıl oynanması gerektiğini açıklayan kısa bir özet yazıdan oluşmaktadır. Oyuna başlamadan önce ekranda görünecektir. Oyuna başladıktan sonra kaybolacaktır.

Oyuna Başlama: Oyun oynama koşullarının hemen altında oyuna nasıl başlandığını açıklayan yazıdır.

Top: Ekrana yuvarlak biçiminde "kırmızı", "yeşil" ve "mavi" renkte olabilir. Ekrana her tıklandığında top zıplamaktadır. Halkalar topun olduğu bölgeden geçerken topun doğru halkadan geçmesi gerekmektedir.

Halka: Her seferinde yukarıdan aşağıya 3 tane renkleri farklı olacak şekilde gelecektir. Topun doğru halkadan geçmesi gerekmektedir.

Menü Tasarımı: En üstte bölümün adının yazdığı Başlık altında ise 3 tane buton bulunmaktadır. Bunlar sonraki bölüme geçme butonu, oyunun tekrar oynandığı buton ve son olarak ise çıkış butonu bulunmaktadır.

Yıldız: Her 10 halkada bir doğru renkteki halkanın içinde gelmektedir. Oyuncu bunu alırsa 1 can eklenmektedir.

4.2. Hazırlık Aşaması

Planlama: Bu aşamada yapılacak olan oyunun süresiyle ilgili hesaplamalar yapılacaktır. Bu planlama yapılırken yapılacak olan geliştirmelerin öncelikleri belirlenmektedir. Sonra da bu geliştirmelerin yapılabilmesi için gerekli süreler belirlenmiştir. Bu seviyeler yüksek, orta, düşük olarak 3 seviye belirlenmiştir. Ayrıca birbirini etkileyen işler ardışık olarak yapılmalıdır.

4.2.1. Koşu gereksinim listesi

Bu liste takımın koşuda gerçekleştireceği işlevlerin bulunduğu listedir [55].

Bu liste takım, ürün sahibi ve ilgili başka kişilere bu koşuda hangi geliştirmeleri yapacakları konusunda bilgi sağlar [55].

Bu listenin özellikleri;

• Bütün ihtiyaçların olduğu listeden koşuda yapılacak işlerin gösterilmesini sağlar.
• Burada bulunan listedeki işlevler sonradan çıkan durumlarda eklenmiş olabilir. Eğer böyle bir durum varsa bunların ihtiyaçların olduğu listeye de eklenmeleri gerekir. Bu kısımda bu işlere süre verilir.
• Görev eğer büyük ise küçük parçalara ayrılırlar. Yani kısaca bir işleve en fazla 16 saat süre verilebilir.
• Ekipteki kişiler işlevleri üzerlerine geçirirler.
• Eğer yeni bir problem oluşmuşsa takımdaki kişiler listeye yeni maddeler açabilmelidirler veya çıkarabilmelidirler.
• Yapılacak özellik kesin veya tam belli değilse zaman tahmini olarak daha büyük zaman verilebilir.

Koşu gereksinim listesi;

Gökyüzü ve bulut görselinin yapılması. Süre 3 Saat'tir. Seviye Yüksek'tir.

Halkaların şeklinin yapılması. Süre 4 Saat'tir. Seviye Yüksek'tir.

Menü Görselleri yapılması. Süre 3 Saat'tir. Seviye Yüksek'tir.

Unity de proje başlangıç ayarların yapılması. Süre 4 Saat'tir. Seviye Yüksek'tir.

Ekranların hem mobil hem de masaüstüne uygun hale getirilmesi. Süre 4 Saat'tir. Seviye Yüksek'tir.

Top'un zıplama hareketinin yapılması. Süre 4 Saat'tir. Seviye Yüksek'tir.

Halkaların hareket ettirilmesi. Süre 4 Saat'tir. Seviye Orta'dır.

Halkaların rastgele şekilde renklendirilmesi. Süre 2 Saat'tir. Seviye Orta'dır.

Topun doğru halkadan geçiş kontrolü yapılması. Süre 7 Saat'tir. Seviye Yüksek'tir.

Oyun Menü tasarımı yapılması. Süre 8 Saattir. Seviye Orta'dır.

Oyunun başlangıç yazılarının oluşturulması. Süre 4 Saat'tir. Seviye Düşük'tür.

Oyun durumu ve can durumu görseli yapılması. Süre 4 Saat'tir. Seviye Düşük'tür.

Oyun'un masaüstünde çıkartılması ve ekran hatalarının düzeltilmesi. Süre 2 Saattir. Seviye Düşük'tür.

Oyun'un iOS ve Android'e çıkartılması ve ekran hatalarının düzeltilmesi. Süre 2 Saattir. Seviye Düşük'tür.

Bu süreler günlük yapılan scrum değerlendirilmesinde bazı problemlerle karşılaşılırsa sürelerde güncelleme yapılabilir.

4.3. Geliştirme

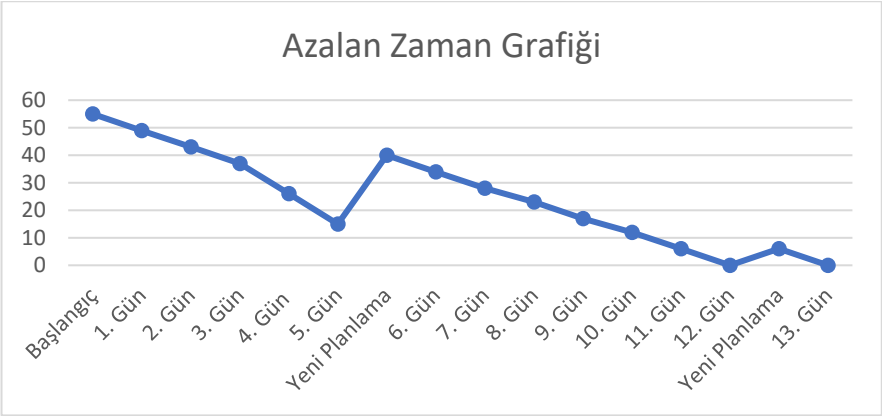
Bu aşamadan sonra Oyun da bazı geliştirmeler öngörülmüştür. Yıldız ekleme oyunun bölümlendirilmesi gibi bu nedenle aslında önceden yapılması gereken maddeler bu aşamadan sonra belirlendiği için bu kısımdan sonra yapılmıştır.

Oyun bölümlendirilmesi yapılması. Süre 12 Saat'tir. Seviye Yüksek'tir.

10 halkada 1 yıldız eklenmesi. Süre 5 Saat'tir. Seviye Yüksek'tir.

Halka hareketlendirilmesi yapılması. Süre 8 Saat'tir. Seviye Yüksek'tir.

Halkaların üç boyutlu yapılması. Süre 6 Saat'tir. Seviye Orta'dır.



Şekil 4.1. Huysuz top oyun planlamasının zaman grafiği

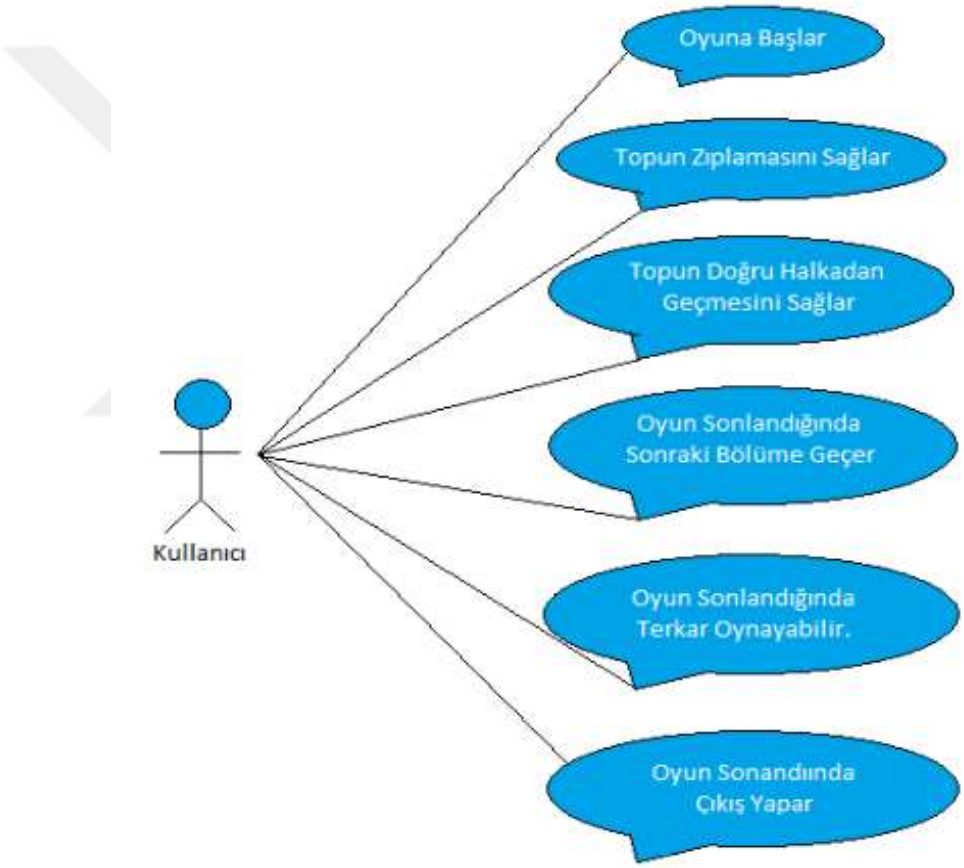
Uygulamaya başlarken toplamda 55 saatlik bir sürede tamamlanacağı planlanmıştır. Ancak çevik yazılımın bize verdiği esneklikten dolayı 5. ve 12. günün sonunda oyunun daha eğlenceli ve oynanabilir olması için birkaç özellik eklenmiştir. Daha sonra koşu

eklenilen özelliklerin önem derecesine göre tekrar planlanmıştır. Uygulamanın zaman grafiği yukarıda Şekil 4.1'de gösterilmiştir.

Oyun oynayacak kişiler göz önüne alınarak oyun üzerinde oyuncuların yapabileceği işlevler kullanıcı diyagramında gösterilmiştir;

Kullanıcı

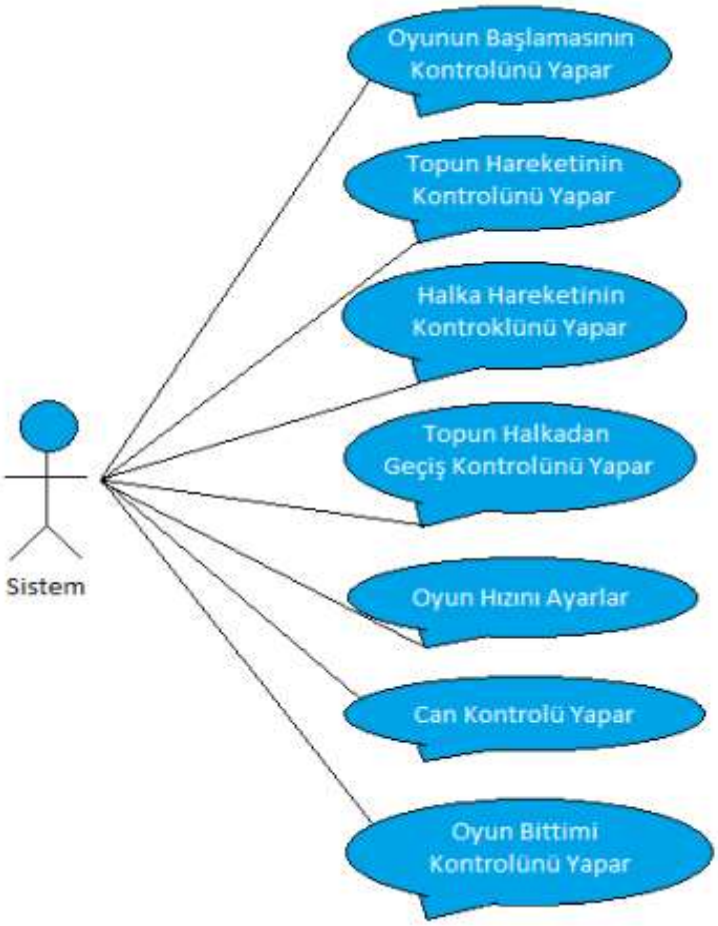
Kullanıcınn oyun oynarken yapabileceği işlemler aşağıda Şekil 4.2'de verilen diyagram ile gösterilmiştir.



Şekil 4.2. Huysuz top oyunu kullanıcının yapabileceği işlemler diyagramı

Sistem

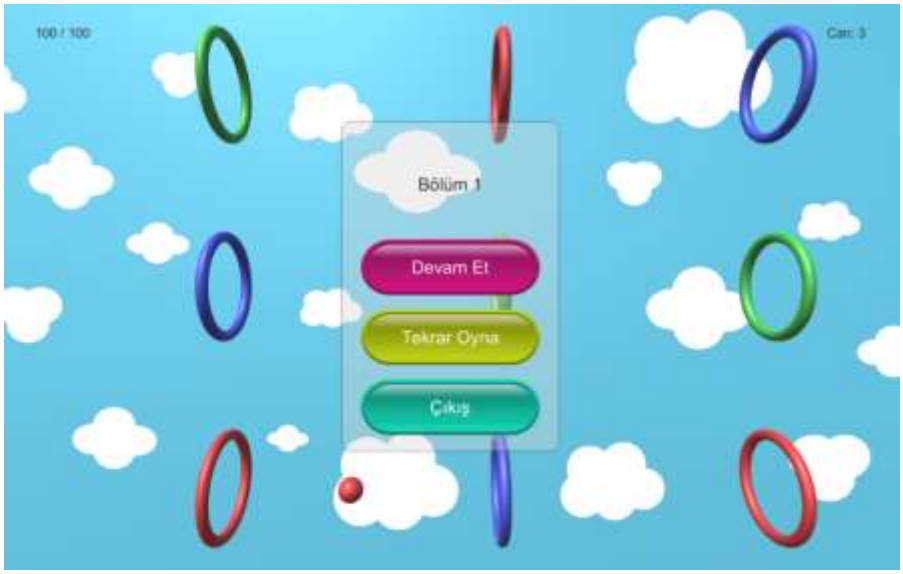
Kullanıcı oyunu oynarken sistemin yaptığı işlemler aşağıda Şekil 4.3'de verilen diyagram ile gösterilmiştir.



Şekil 4.3. Huysuz top oyunu sistemin yaptığı işlemler diyagramı

Aşağıdaki görseller oyunumuzdan alınan ekranlardaki görüntülerden oluşmaktadırlar;

Şekil 4.4. Huysuz top oyun başlangıç ekranı



Şekil 4.5. Huysuz top bölüm sonu ekranı

Şekil 4.6. Huysuz top oyun bitiş ekranı

5. BALON VURMA

Yapacağımız uygulamamızda ilk oyunda olduğu gibi Scrum metodu kullanılarak oyunun başlangıçta verilen süreyle bitişte verilen sürenin karşılaştırılması yapılacaktır. Aşağıda yaptığımız uygulamanın aşamalarını açıklıyoruz. Daha sonra çıkan verilere göre sonuçları inceleyeceğiz.

5.1. Üretim Öncesi

Bu aşamada oyunun temel özellikleri belirlenmiştir. Oyunumuz bir ok ve bir balondan oluşmaktadır. Kullanıcı ok ile balonu vurmaya çalışacaktır.

Oyunumuz Unity platformu kullanılarak C# dilinde kodlanacaktır. Masaüstü, Android ve iOS platformlarında oynanabilecek şekilde yapılacaktır. Prototip üretimi yapılmamıştır. Direk oyunun yapımı işlemine geçilmiştir. Çevik yöntemin başarısının daha iyi test edilebilmesi için oyunda gerekli görünen yerdeki değişiklikler fark edildikten sonra planlanarak değiştirilecektir.

Geliştirme aşamasında oyunumuz test edildikçe bazı değişiklikler yapılmıştır. Bunlar ilk önce balonun sağa ve sola doğru biraz açısal olarak hareket etmesi sağlanmıştır. Bu şekilde ekrana bir canlılık getirilmiştir. Daha sonra oyunun kolay olduğu fark edildiği için balonun hareket etmesine karar verdik. Bu değişiklikten sonra da her 5 balon da bir kullanıcıya bir can verilmesine karar verilmiştir.

5.1.1. Oyun tasarım dokümantasyonu

Bu oldukça önemli bir konudur. Çünkü tasarımda bir şey atlanırsa veya ihmal edilirse sonradan oyunumuzdaki bazı özelliklerin kaybolmasına neden olabilir. Bu durum oyunun en önemli özelliği olan “eğlence” faktörünün azalmasına sebep olur.

5.1.2. Ürün gereksinim listesi

Gökyüzü: Arka planda gökyüzü görseli planlanmıştır.

Bulut: Arka plan ile iç içedir. Ekranda rasgele gelecek şekilde bulut görselleri yapılmalıdır.

Oyunun durumu: Ekranda kaç tane balon patlatıldığını kullanıcının görebilmesi için ekranın sol üst köşesinde bu görsel olacaktır.

Can: Oyunda kaç canımızın kaldığını gösterir. Ekranın sağ üst köşesinde olacaktır.

Oyun Oynanma Koşulları: Oyunun nasıl oynanması gerektiğini açıklayan kısa bir özet yazıdan oluşmaktadır. Oyuna başlamadan önce ekranda görünecektir. Oyuna başladıktan sonra kaybolacaktır.

Oyuna Başlama: Oyun oynama koşullarının hemen altında oyuna nasıl başlandığını açıklayan yazıdır.

Ok: Ekranın altında olacaktır. Kullanıcı balona doğru fırlatabilecektir.

Balon: Ekranın herhangi bir yerinde çıkabilir. Açısal hareket edebilir. İlerleyen zamanlarda konum olarak da hareket edebilir.

Menü Tasarımı: En üstte bölümün adının yazdığı Başlık altında ise 3 tane buton bulunmaktadır. Bunlar oyuna devam etme, oyunun tekrar oynandığı buton ve son olarak ise çıkış butonu bulunmaktadır.

Bir ikonu: Her 5 balonda bir balonun içinde gelmektedir. Oyuncu bunu alırsa 1 can eklenmektedir.

5.2. Hazırlık Aşaması

Planlama: Bu aşamada yapılacak olan oyunun süresiyle ilgili hesaplamalar yapılacaktır. Bu planlama yapılırken yapılacak olan geliştirmelerin öncelikleri belirlenmektedir. Sonra da bu geliştirmelerin yapılabilmesi için gerekli süreler belirlenmiştir. Bu seviyeler yüksek, orta, düşük olarak 3 seviye belirlenmiştir. Ayrıca birbirini etkileyen işler ardışık olarak yapılmalıdır.

5.2.1. Koşu gereksinim listesi

Koşu Gereksinim Listesi ile ilgili bilgiler ilk oyunun bu bölümünde verilmiştir. Burada sadece liste paylaşılacaktır.

Gökyüzü ve bulut görselinin yapılması. Süre 3 Saat'tir. Seviye Yüksek'tir.

Balon şeklinin yapılması. Süre 2 Saat'tir. Seviye Yüksek'tir.

Menü Görselleri yapılması. Süre 2 Saat'tir. Seviye Yüksek'tir.

Unity de proje başlangıç ayarların yapılması. Süre 4 Saat'tir. Seviye Yüksek'tir.

Ekranların hem mobil hem de masaüstüne uygun hale getirilmesi. Süre 4 Saat'tir. Seviye Yüksek'tir.

Ok'un açısal hareketinin yapılması. Süre 2 Saat'tir. Seviye Yüksek'tir.

Ok'un fırlatma hareketinin yapılması. Süre 3 Saat'tir. Seviye Yüksek'tir.

Balon 'un rastgele ekranda belirmesi. Süre 2 Saat'tir. Seviye Orta'dır.

Balon 'un patlama animasyonu. Süre 3 Saat'tir. Seviye Yüksek'tir.

Ok'un Balonu vurup vurmadığı kontrolü yapılması. Süre 5 Saat'tir. Seviye Yüksek'tir.

Oyun Menü tasarımı yapılması. Süre 6 Saattir. Seviye Orta'dır.

Oyunun başlangıç yazılarının oluşturulması. Süre 2 Saat'tir. Seviye Düşük 'tür.

Oyun durumu ve can durumu görseli yapılması. Süre 3 Saat'tir. Seviye Düşük 'tür.

Oyun' un masaüstünde çıkartılması ve ekran hatalarının düzeltilmesi. Süre 2 Saattir. Seviye Düşük' tür.

Oyun'un iOS ve Android'e çıkartılması ve ekran hatalarının düzeltilmesi. Süre 2 Saattir. Seviye Düşük' tür.

Bu süreler günlük yapılan scrum değerlendirilmesinde bazı problemlerle karşılaşılırsa sürelerde güncelleme yapılabilir.

5.3. Geliştirme

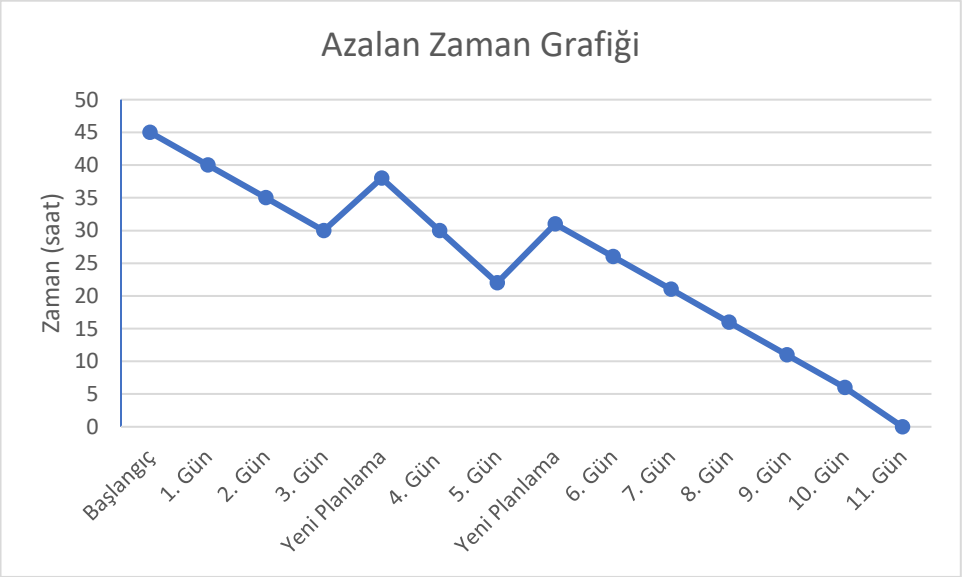
Bu aşamadan sonra Oyun da bazı geliştirmeler öngörülmüştür. Bir ikonu ekleme balonun hareket etmesi gibi bu nedenle aslında önceden yapılması gereken maddeler bu aşamadan sonra belirlendiği için bu kısımdan sonra yapılmıştır.

Balon ’un açısal hareketinin yapılması. Süre 3 Saat’tir. Seviye Orta’dır.

Balon’ un konumsal hareketinin yapılması. Süre 5 Saat’tir. Seviye Orta’dır.

Ok Balonu vurduğunda bir ikonunun eklenmesi. Süre 6 Saat’tir. Seviye Yüksek’tir.

Bir ikonunun animasyonu. Süre 3 Saat’tir. Seviye Yüksek’tir.



Şekil 5.1. Balon vurma oyun planlamasının zaman grafiği

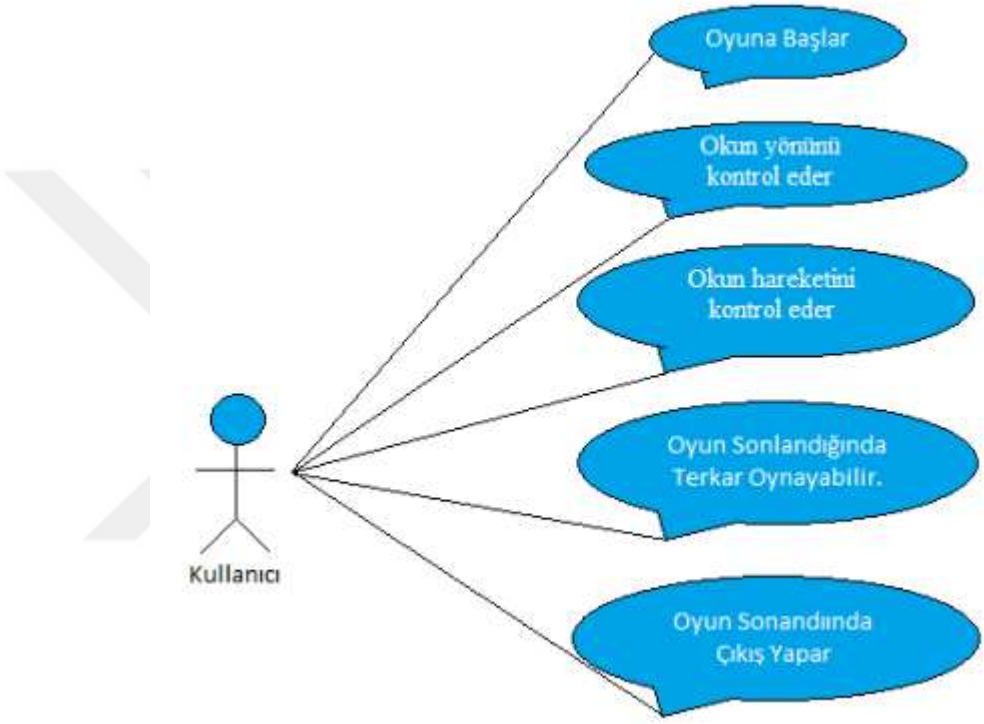
Uygulamaya başlarken toplamda 45 saatlik bir sürede tamamlanacağı planlanmıştır. Ancak Çevik yazılımın bize verdiği esneklikten dolayı 3. ve 5. günün sonunda oyunun daha eğlenceli ve oynanabilir olması için birkaç özellik eklenmiştir. Daha sonra koşu eklenilen özelliklerin önem derecesine göre tekrar planlanmıştır. Uygulamanın zaman grafiği yukarıda Şekil 5.1’de gösterilmiştir.

Aşağıdaki kısımda yaptığımız oyunun daha iyi anlaşılabilmesi için kullanıcı ve sistem senaryolarını şekil olarak gösterimi yapılmıştır.

Yaptığımız şekillerden ilkinde oyunu oynayacak kişiler göz önüne alınarak oyun üzerinde oyuncuların yapabileceği işlevler kullanıcı diyagramında gösterilmiştir;

Kullanıcı

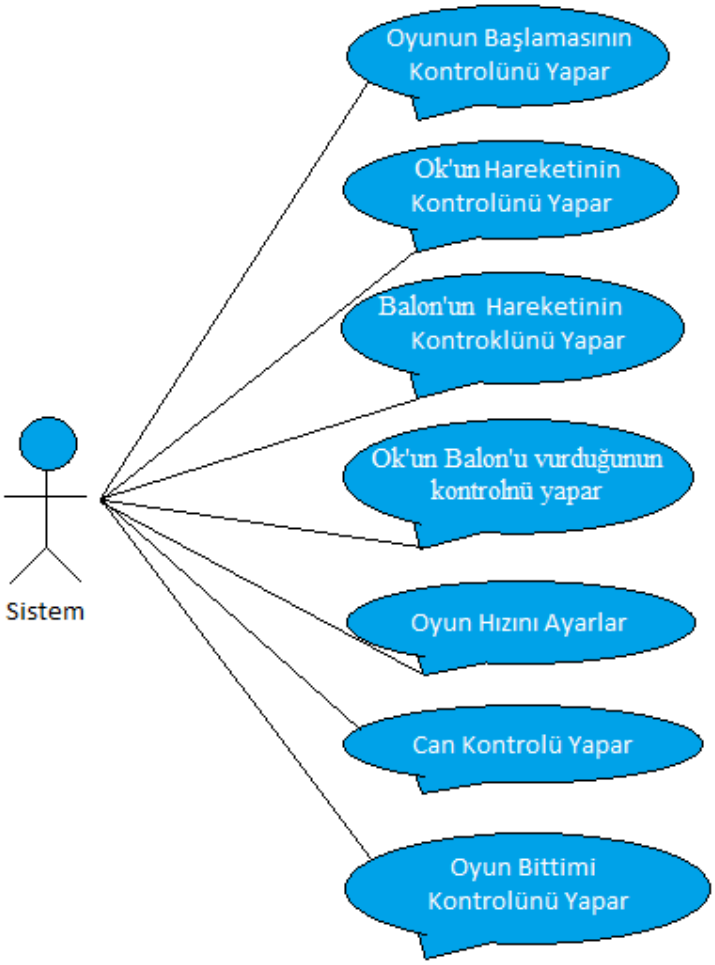
Kullanıcınn oyun oynarken yapabileceği işlemler aşağıda Şekil 5.2'de verilen diyagram ile gösterilmiştir.



Şekil 5.2. Balon vurma oyunu kullanıcının yapabileceği işlemler diyagramı

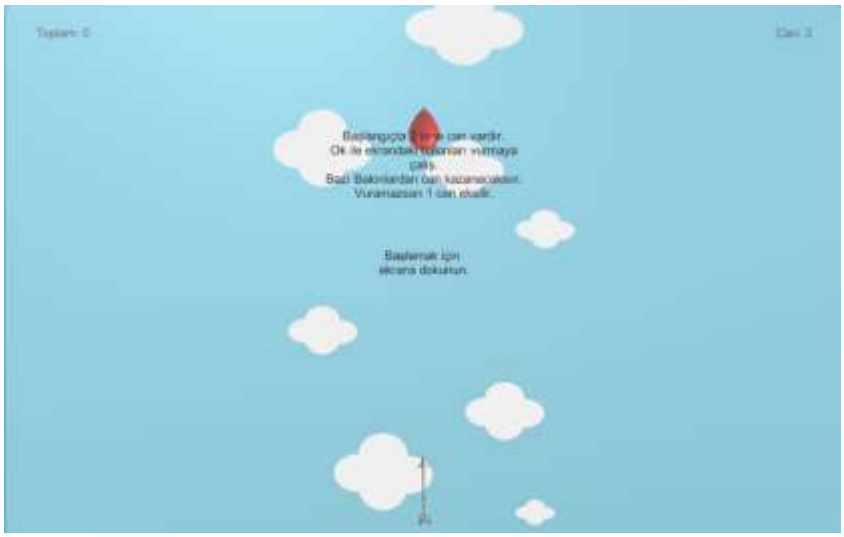
Sistem

Kullanıcı oyunu oynarken sistemin yaptığı işlemler aşağıda Şekil 5.3'de verilen diyagram ile gösterilmiştir.

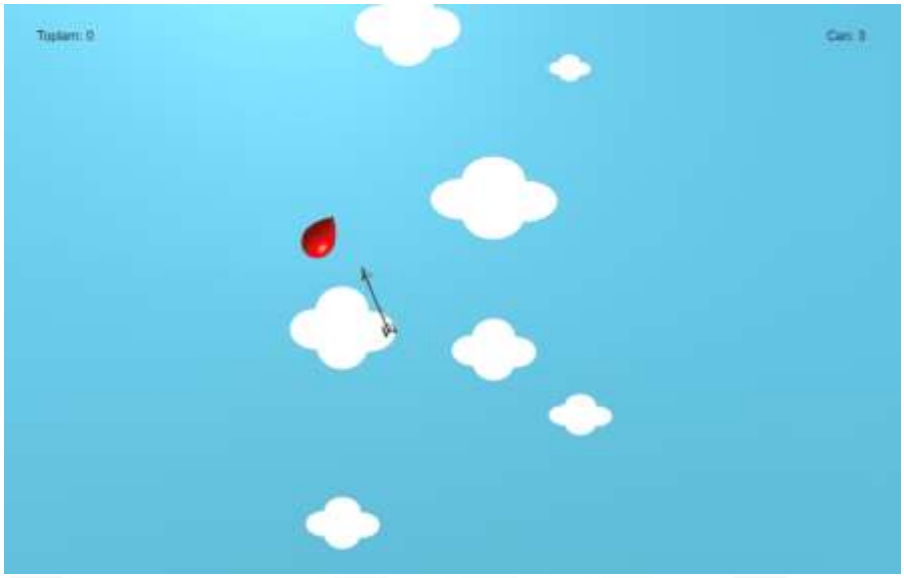


Şekil 5.3. Balon vurma oyunu sistemin yaptığı işlemler diyagramı

Uygulamadan alınan ekran görüntüleri aşağıda görüntülenmektedir;



Şekil 5.4. Balon vurma oyunu başlangıç ekranı



Şekil 5.5.Balon vurma oyunu ok'u fırlatırken oyun ekranı

Şekil 5.6. Balon vurma oyun bitiş ekranı

6. SONUÇLAR VE ÖNERİLER

Çevik yöntemlerde proje esnek olduğu için bitiş tarihi belirlemek oldukça zordur. Yaptığımız oyunlarda proje planlanırken verdiğimiz toplam süre ile proje bitimindeki ortaya çıkan süre arasındaki farkı ve bu farkın ne tür içeriklerden oluştuğuna bakarak, çevik yöntem kullanılırken ortalama bir bitiş süresi hesaplanması amaçlanmaktadır.

Huysuz top başlarken toplamda 55 saatlik bir süre sonra projenin biteceği planlanmıştır. Ancak daha sonra proje geliştirildiği sürece ortaya çıkan ürün incelendikçe oyunun daha iyi, eğlenceli ve oynanabilir olması için bazı değişiklikler yapılarak toplam 86 saatlik sürenin sonunda sonlanmıştır. Projemizde çevik yöntemin esneklik ilkesinden dolayı bitiş süresinde yaklaşık 31 saatlik bir sapma meydana gelmiştir.

Balon vurma oyununda ise başlarken 45 saatlik bir süre sonra projenin biteceği planlanmıştır. Daha sonra oyundaki değişiklikler sonunda 17 saatlik bir sapma meydana gelmiştir. Toplamda 62 saatlik sürenin sonunda sonlanmıştır.

Her iki oyundaki sapma değeri toplam süre baz alındığında oldukça fazla bir süredir.

Bu sapmanın nedenlerini inceleyelim:

Burada yapılan oyunu genellediğimizde 4 ana bölümden oluştuğunu söyleyebiliriz. Bu bölümler şunlaradan oluşmaktadır.

• Oyun görsellerinin tasarımı.
• Oyundaki menüler ve oyun bilgilerinin yer aldığı ekranlar.
• Oyun ekranında oyunun daha önemsiz yan karakterlerinin oluşturduğu görseller ve senaryolar.
• Oyun ekranında oyunun ana karakterlerinin oluşturduğu görseller ve senaryolar.

6.1. Oyun Görsellerinin Tasarımı

Huysuz top oyun planlama aşamasında bu kısım için toplamda 14 saatlik bir süre hesaplanmıştır. Oyun ilerledikçe oyunda yeni özellikler ekleniyor fakat bu eklenen özellikler genel olarak var olan senaryoların geliştirilmesi üzerine olmuştur. Oyuna sonradan 10 halkada bir can ekleme özelliği yapıldığı için yıldız tasarımına gerek duyulmuştur. Ayrıca halkaların 3 boyutlu olarak tekrar tasarlanmasına gerek duyulmuştur. Bu iki madde için toplam da 6 saatlik ek süreye ihtiyaç duyulmuştur. Yani toplamda 20 saatlik sürenin 14 saati oyun planlama aşamasında tespit edilmiştir. Yani %43'luk bir zaman sapması yaşanmıştır.

Balon vurma oyununda ise kullanıcının can kazanması için 1 ikonu tasarlanmıştır. Bu tasarım için 3 saatlik ek süreye ihtiyaç duyulmuştur. Toplamda 14 sürenin 11 saati oyun planlama aşamasında tespit edilmiştir. Yani %27'lik bir zaman sapması yaşanmıştır.

İki oyunu karşılaştırdığımızda birinde %43 diğerinde %27'lik bir süre sapması olmuştur.Ortalama ise %35'lik bir süre sapması olmuştur.

6.2. Oyun Menüleri ve Bilgileri

Huysuz top oyunu planlaması yapılırken bu kısım için toplamda 16 saatlik bir süre hesaplanmıştır. Oyuna yeni özellikler eklendikçe bu özellikleri ekranda göstermek gerekmektedir. Bu yüzden yeni menü ve metinler hazırlanması gerekti. Bu oyun özelinde ise kullanıcı her yandığında baştan başlamak zorunda kalmaktaydı. Bu durum oyunumuz oynanabilirliğini kötü yönde etkilemekteydi. Oyunun eğlence yönünü azaltmaktaydı. Bu nedenle oyuna can özelliği eklenmiştir. Bu görselin gösterilmesi ve can ekleme, kullanıcı yandığında can silme gibi güncellemeler gerektiği için toplamda 2 saatlik süre fazladan tespit edilmiştir. Toplamda 18 saatlik sürenin 16 saati oyun planlama aşamasında tespit edilmiştir. Yani %13'lik bir zaman sapması olmuştur.

Balon vurma oyununda ise geliştirilirken herhangi bir değişkliğe ihtiyaç duyulmamıltır.Toplamda 14 saatlik sürenin tamamı oyun başında planlanmıştır. Zaman sapması %0'dır.

İki oyunu karşılaştırdığımızda birinde %13 diğerinde %0'lık bir süre sapması olmuştur.Ortalama ise %7'lik bir süre sapması olmuştur.

6.3. Oyun Yan Karakterleri Yapımı

Huysuz top oyunu planlamasında bu kısım için 10 saatlik bir süre hesaplanmıştır. Oyun geliştikçe yapılan testler sonucu oyunun iyileştirilmesi için her 10 halkada bir doğru renkteki halkanın içine yıldız görseli eklenmiştir. Oyuncu bu halkanın içinden geçince yıldız kayboluyor ve oyuncuya bir can daha ekleniyor. Ayrıca halkaların 3 boyutlu yapılmasında da bu bölüm için zamana ihtiyaç duyulmuştur. Bu özelliklerin yapılması için toplamda 3 saatlik bir süre harcanmıştır. Toplamda 13 saatlik sürenin 10 saati oyun planlama aşamasında tespit edilmiştir. Yani yaklaşık %30 luk bir zaman sapması olmuştur.

Balon vurma oyununda ise başlangıçta 4 saatlik bir süre planlanmıştır. Fakat oyunu geliştirirken Oyuna gerçeklik katmak için ekrandaki balonu açısal hareket ettirmek istedik. Bunun için ise 3 saat'lik bir ek süre planlanmıştır. Toplamda 7 saatlik süre harcanmıştır. Yani yaklaşık %75'lik bir süre sapması olmuştur.

İki oyunu karşılaştırdığımızda birinde %30 diğerinde %75'lik bir süre sapması olmuştur. Ortalama ise %53'lik bir süre sapması olmuştur. İkinci oyunda süre sapmasının daha çok olmasının sebebi aslında oyun sonunda farkettiğimiz balon karakterinin bazı özellikleri aslında oyunda ön plana çıkarak ana karakter grubuna daha yaklaşmasından kaynaklanmıştır.

6.4. Oyun Ana Karakterleri Yapımı

Huysuz top oyunu planlamasında bu kısım için 11 saatlik bir süre hesaplanmıştır. Oyun geliştikçe ortaya bazı sorunlar çıkmıştır. Halkalar sabit bir şekilde duruyor. Topun rengi sabit ve sadece bir bölümden oluşan bir oyun olması gibi konular oyunu oldukça sıkıcı bir hale getirdiği düşünülmüştür. Bu nedenle oyun toplamda 4 bölüm haline getirilmiştir. Bu bölümlerde topun renginin rastgele değişmesi halkaların hareketli olması gibi oyunu monotonluktan çıkaracak ve daha eğlenceli hale getirecek köklü değişiklikler planlanmıştır. Bu özelliklerin yapılması için toplamda 20 saatlik bir ek süreye daha ihtiyaç duyulmuştur. Sonuç olarak toplamda 31 saatlik sürenin 11

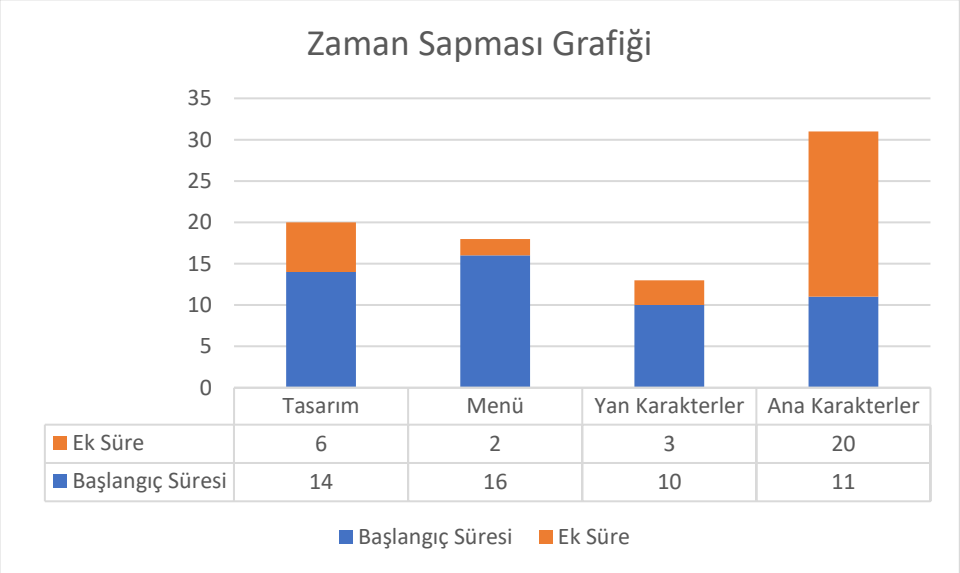
saati oyun başlangıcında planlanmıştır. Yani yaklaşık %180'lik bir zaman sapması olmuştur.

Balon vurma oyunu planlamasında ise 11 saatlik bir süre planlaması yapılmıştır. Ancak daha sonra oyunu eğlenceli yapmak için can ekleneceği zaman balonun içine 1 ikonunun eklenmesi, balon vurulduğunda bir ikonunun anmasyonu ve son olarak da balonun konumsal hareketi yapılması planlanmıştır. Bunlar için ise 11 saatlik bir ek süre planlanmıştır. Yani toplamda 22 saatlik bir süre harcanmıştır. Zaman sapması ise %100 olmuştur.

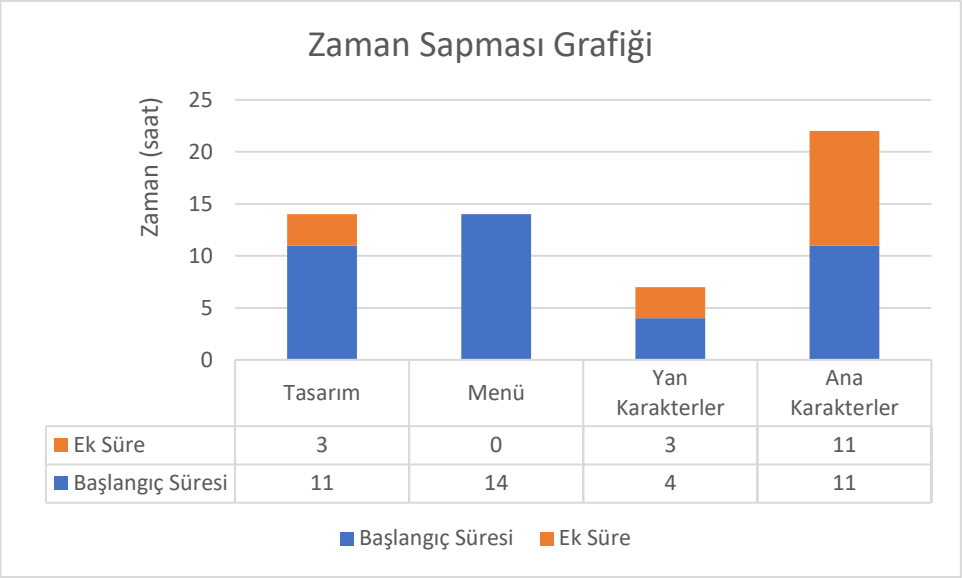
İki oyunu karşılaştırdığımızda birinde %180 diğerinde %100'lik bir süre sapması olmuştur.Ortalama ise %140'lik bir süre sapması olmuştur. Bir önceki bölümde bahsettiğimiz gibi ikinci oyunda bu oranın daha düşük olmasının sebebi yan karakter ve ana karrakterlerin birbirine yakın olmasıdır. Geliştirme yapılırken yan karakterlerin ana karakter grubuna daha yakınlaşmasından kaynaklanmaktadır.

6.5. Bulgular ve Öneriler

Yukarıda belirtilen başlangıç ve daha sonradan değişen koşullara göre zaman sapması grafiği aşağıda Şekil 6.1 ve Şekil 6.2'de gösterilmiştir.

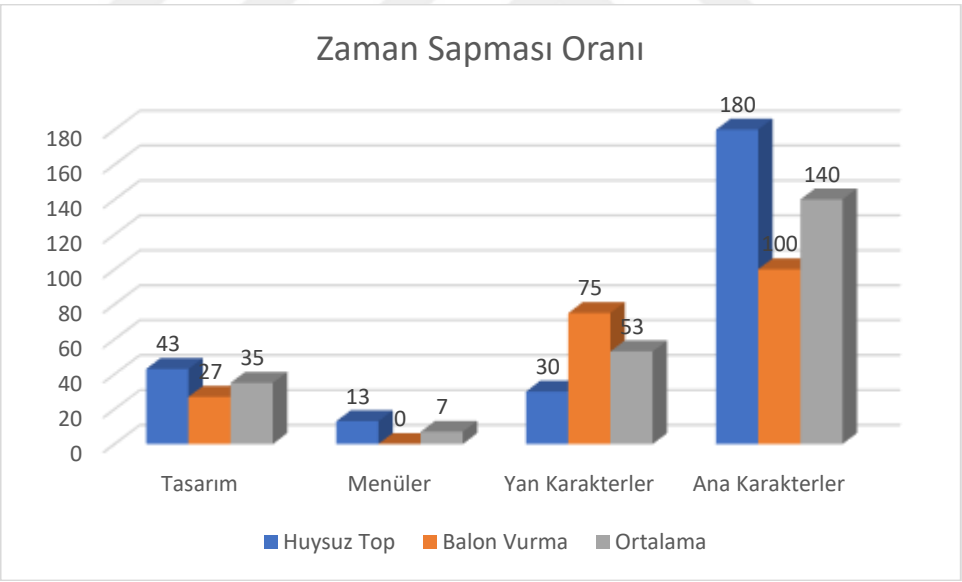


Şekil 6.1. Huysuz Top oyununun zaman sapması grafiği



Şekil 6.2. Balon vurma oyununun zaman sapması grafiği

Yapılan iki oyunun süre sapmalarının karşılaştırıldığı grafik ise Şekil 6.3'de gösterlmiştir.



Şekil 6.3. İki oyunun zaman sapması karşılaştırma grafiği

Bir diğer olarak da proje başlangıç ayarlarının yapılması için 4 saatlik bir süre planlanmıştır. Bu süreyi hesaba katmamamızın sebebi bu tamamen arka planda kodu geliştiren kişilerin hızına göre planlanan bir süredir. Bu nedenle bu sürede herhangi bir sapma olmayacağı için bu kısım süre hasaplamasında sadece proje planlamasının sonuna eklenir.

Yukarıdaki 4 ana bölüm incelendiğinde oyun tasarımı, oyun menüleri tasarımı ve oyun yan karakterlerinin yapılması çok kısıtlı oranda değişirken oyunun senaryosunu, kurgusunu ve oynanma durumlarını doğrudan etkileyen bölümlerde bu oran oldukça yüksektir. Sonuç olarak oyun başlangıcında planlanan sürelere yukarıda belirttiğimiz oranlarda süre eklenirse proje için bir bitiş tarihi hesaplanabilir. Fakat çevik yöntemlerin esneklik ilkesinden dolayı yinede kesin bir tarih çıkarılması kestirilemez. Sadece tahmin edilebilir. Bu örnekler çoğaltıldıkça daha isabetli bir sonuca varılma ihtimali daha da yükselecektir.

Burada bu sürenin çıkarılmasını olumsuz yönde etkileyecek en önemli kısım yukarıda belirtilen 4 ana bölümün doğru şekilde tespit edilememesi olacaktır. Özellikle oyunun ana karakterleri ve daha önemsiz yan ve arka karakterlerinin olduğu bölümlerin toplam süre tespitinin birbirinden iyi şekilde ayrılması gerekir. Bu ayrımı yapan kişilerin oyunun senaryosunu çok iyi anlamış olması gerekir.

Bu önermeyi olumsuz yönde etkileyecek bir husuda şudur.

Bazen oyunun yan karakterleri o kadar beğenilirki oyun üzerindeki etkisi arttırılarak oyunun ana karakterleri haline gelebilir. Bu da süre tahminini olumsuz yönde etkileyebilir. Bu nedenle oyunun yan karakterlerine karar verilirken dikkat etmeliyiz.

Ortaya çıkan sonuçlara baktığımızda oyundaki yan karakterler ile ana karakterlerin olduğu bölümlerin süre sapması oldukça fazladır.

Bu konuda yapılacak diğer çalışmalarda daha iyi bir bitiş tarihi tespiti yapılabilmesi için kullandığımız 4 ana bölüm detaylandırılabilir. Bu bölümler ne kadar detaylı olursa süre sapmalarının hesabı o kadar doğru tespit edilir.

KAYNAKLAR

[1] Baytam V., Kalıpsız O., Scrum Yazılım Geliştirme Modeli Yönetim Aracı ScrumMApp, Beşinci Ulusal Yazılım Mühendisliği Sempozyumu, Ankara, 26-28 Eylül 2011.

[2] Highsmith J., What is Agile Software Development, STSC Crosstalk. Journal of Defense Software Engineering, 2002, **15**(10), 4-10.

[3] Grenning J., Launching Extreme Programming at a Process-Intensive Company. IEEE Software, 2001, **18**(6), 27-33.

[4] Lippert M., Becker-Pecbau P., Breitling H., Koch J., Kornstadt A., Roock S., Zullighoven H., Developing Complex Projects Using XP with Extensions, Computer, 2003, **36**(6), 67-73.

[5] Reifer D. J., Maurer F., Erdogmus H., Scaling Agile Methods, IEEE Software, 2003, **20**(4), 12-14.

[6] Beck K., Beedle M., Van B. A., Cockburn A., Cunningham W., Fowler M., Kern J., Manifesto for Agile Software Development, Agile Esentials, https://www.agilealliance.org/agile101/the-agile-manifesto/, (Ziyaret tarihi: 18 Eylül 2019).

[7] Süloğlu S., Yöntem Çevik Olunca, 2. Ulusal Yazılım Mühendisliği Sempozyumu, Ankara, 26 Eylül 2005.

[8] Duru I., Çevik Yöntemlerde Mobil Uygulama Tasarımı ve Gerçekleştirilmesi, Yüksek Lisans Tezi, Yıldız Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, 2014, 364133.

[9] Vlaanderen K., Jansen S., Brinkkemper S., Jaspers E., The Agile Requirements Refinery Applying Scrum Principles to Software Product Management, Information and Software Technology, 2011, **53**(1), 58-70.

[10] Şenkaya E., Yazılım Projelerinde Başarı Anahtarları, CIO Club Bilişim Dergisi, 2009, 54-57.

[11] Cobb M., Unfinished Voyages a Follow-Up to The Chaos Report, Integrated Computer Systems, http://www.ics-support.com/download/StandishGroup _CHAOSReport.pdf, (Ziyaret tarihi: 21 Eylül 2019).

[12] Johnson, M., Shine Technologies Agile Methodologies Survey Results, Shine Technologies, http://www.shinetech, (Ziyaret tarihi: 13 Mart 2020).

[13] Cockburn A., Highsmith J., Agile Software Development The People Factor, Computer, 2001, **34**(11), 131-133.

[14] Karlıdere T., Kalıpsız O., Yazılım Mühendisliği Projelerinde Çevik Yaklaşımların Yeri, Ulusal Yazılım Mühendisliği Sempozyumu, Ege Üniversitesi İzmir, 23-25 Eylül 2003.

[15] Tekinerdogan B., Formalizing Agile Software Development Methods, Impact of Software Process on Quality Workshop, Ankara, 6 Haziran 2003

[16] Highsmith J., Cockburn A., Agile Software Development The Business of Innovation, Computer, 2001, **34**(9), 120-127.

[17] Sahin E., Keskin I., Koç H., CMMI-DEV Seviye-3 Sertifikasyonuna Sahip Bir Organizasyonda Scrum Çevik Yazılım Geliştirme Yönteminin Yazılım Geliştirme Çalışmalarında Uygulanması, Ulusal Yazılım Mühendisliği Sempozyumu (UYMS), İzmir, 26 Eylül 2013.

[18] Wijesinghe R., Role of Software Architecture in Agile Software Development, Ruwan.Net, http://ruwandotnet.wordpress.com, (Ziyaret tarihi: 14 Şubat 2020).

[19] Boehm B., Get Ready for Agile Methods with Care, Computer, Institute of Electrical and Electronics Engineers (IEEE), 2002, **35**(1), 64-69.

[20] Lindvall M., Basili V., Boehm B., Costa P., Dangle K., Shull F., Zelkowitz M., Empirical Findings in Agile Methods, Conference on Extreme Programming and Agile Methods, Berlin, 4-7 Ağustos 2002.

[21] Cohen D., Lindvall M., Costa P., Agile Software Development, Dacs Soar Report, 53-58, 2003.

[22] McDowell C., Werner L., Bullock H. E., Fernald J., The Impact of Pair Programming on Student Performance Perception and Persistence, 25th International Conference on Software Engineering, Portland or ABD, 3-10 Mayıs 2003.

[23] Wood W. A., Kleb W. L., Exploring XP for Scientific Research, IEEE Software, 2003, **20**(3), 30-36.

[24] Rasmussen J., Introducing XP Into Greenfield Projects, Lessons Learned IEEE Software, 2003, **20**(3), 21-28.

[25] Murru O., Deias R., Mugheddue G., Assessing XP, European Internet Company IEEE Software, 2003, **20**(3), 37-43.

[26] Abrahamsson P., Warsta J., Siponen M. T., & Ronkainen J., New Directions on Agile Methods: a Comparative Analysis, 25th International Conference on Software Engineering, Portland or ABD, 3-10 Mayıs 2003.

[27] Turk D., France R., Rumpe B., Limitations of Agile Software Processes, Cornell University, https://arxiv.org/ftp/arxiv/papers/1409/1409.6600.pdf, (Ziyaret tarihi: 11 Haziran 2020).

[28] Williams L., Cockburn A., Agile Software Development It's About Feedback and Change, IEEE Computer, 2003, **36**(6), 39-43.

[29] Abrahamsson P., Salo O., Ronkainen J., Warsta J., Agile Software Development Methods Review and Analysis, Cornell University https://arxiv.org/ftp/arxiv/papers/1709/1709.08439.pdf, (Ziyaret tarihi: 16 Ağustos 2020).

[30] Riehle D., A Comparison of The Value Systems of Adaptive Software Development and Extreme Programming How Methodologies May Learn From Each Other, First International Conference on Extreme Programming and Flexible Processes in Software Engineering, Cagliari Italy , 21-23 Haziran 2000.

[31] Schuytema P., Design de Games, Uma Abordagem Prática, Pioneira Brasil, 1 Ocak 2008.

[32] Wells D., Extreme Programming a Gentle Introduction, Agile Process, www.extremeprogramming.org, (Ziyaret tarihi: 28 Mart 2020).

[33] Fowler M., The New Methodology, Wuhan University Journal of Natural Sciences, 2001, **6**(1-2), 2-12

[34] Keith E. R., Agile Software Development Processes a Different Approach to Software Design, Agile Esentials, http://cf.agilealliance.org/articles/system /article/file /1099/file.pdf, (Ziyaret tarihi: 21 Eylül 2019).

[35] Cockburn A., Highsmith J., Johansen K., Jones M., Crystal Methodologies, Datasel Bilgi Sistemleri A.Ş, https://www.emo.org.tr/ekler/401bb41f7e78637 _ek.pdf, (Ziyaret tarihi: 4 Ekim 2019).

[36] Cohn M. W. Selecting an Agile Process Choosing Among the Leading Alternatives, Proceeding of SD Best Pratices Conference & Expo 2004, Chicago, 21-24 Eylül 2004.

[37] Neubulon Pty. Ltd., Feature Driven Development (FDD), Neubulon Pty. Ltd, http://www.featuredrivendevelopment.com, (Ziyaret tarihi: 14 Aralık 2019).

[38] Landaeta R. E., Viscardi S., Tolk A., Strategic Management of Scrum Projects an Organizational Learning Perspective, First International Technology Management Conference, San Jose CA USA, 27-30 Haziran 2011.

[39] Takeuchi H., Nonaka I., The New Product Development Game, Harvard Business Review, 1986, **64**(1), 137-146.

[40] Highsmith J., Agile Methodologies: Problems, Principles and Practices, XP2001 Conference, Villasimius, Sardinia, Italy, 20-23 Mayıs 2001.

[41] Guang-yong H., Study and Practice of Import Scrum Agile Software Development, 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, China, 27-29 Mayıs 2011.

[42] Wikia, Video Game Industry, Wikia, http://vgsales. wikia.com/wiki/Video_game_industry (Ziyaret tarihi: 38 Haziran 2020).

[43] Godoy A., Barbosa E. F., Game-Scrum an Approach to Agile Game Development, Proceedings of SBGames, Sao Carlos (SP) Brazil, 8-10 Kasım 2010.

[44] Gregory D., Building a Mindset for Rapid Iteratioon Part1 The Problem, Gamasutra, http://www.gamasutra.com/view/feature/3645/, (Ziyaret tarihi: 18 Ekim 2019).

[45] Petrillo F., Pimenta M., Trindade F., Dietrich C., We Have a Problem a Survey of Actual Problems in Computer Games Development, Acm Symposium on Applied Computing, Fortaleza Ceara, 16-20 Mart 2008.

[46] Kanode C. M., Haddad H. M., Software Engineering Challenges in Game Development, Sixth International Conference on Information Technology: New Generations, Las Vegas Nevada USA, 27-29 Nisan 2009.

[47] Demachy T., Extreme Game Development: Right on Time, Every Time, Gamasutra, https://www.gamasutra.com/view/feature/131236/extreme_ game_development_right_on_.php, (Ziyaret tarihi: 8 Ekim 2019).

[48] Schwaber K., Sutherland J., Scrum Guide Developed and Sustained, Scrum Org, https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf, (Ziyaret tarihi: 23 Mart 2020).

[49] Keith C, Kanban for Video Game Development, InfoQ, http://www.infoq.com/presentations/kanban-video-game-dev, (Ziyaret tarihi: 16 Ekim 2019).

[50] Myllyaho M., Salo O., Kääriäinen J., Hyysalo J., Koskela J., A Review of Small and Large Post-Mortem Analysis Methods, Proceedings of the ICSSEA, Paris, 1-8 Aralık 2004.

[51] Sutherland J., Agile Development: Lessons Learned From the First Scrum Cutter Agile Project Management Advisory Service, Executive Update, 2004, **5**(20), 1-4.

[52] Sutherland J. V., Schwaber K., The Scrum Methodology, Business Object Design and Implementation OOPSLA Workshop, Texas, 16 Ekim 1995.

[53] Schwaber, K., Scrum Development Process, Business Object Design and Implementation, London, 28 Nisan 1997.

[54] Hundermark P., Do Better Scrum, ScrumSense, http://www.scrumsense. com/wp- content/uploads/2009/12/DoBetterScrumv2.pdf, (Ziyaret tarihi: 24 Ekim 2019).

[55] Sutherland J., Schwaber K., The Scrum Papers: Nuts, Bolts and Origins of an Agile Process, England, 14 Ekim 2007.

[56] Ambler S., Answering the Where is the Proof That Agile Methods Work? Question, Agile Modeling, http://agilemodeling.com/essays/proof.htm (Ziyaret tarihi: 12 Mart 2020).

[57] Boehm B., Turner R., Using Risk to Balance Agile and Plan-Driven Methods, Computer, Institute of Electrical and Electronics Engineers (IEEE), 2003, **36**(6), 57-66.

KİŞİSEL YAYIN VE ESERLER

[1] **Mercan Ş.**, Becerikli Y., Agile Methods in Game Programming Based on Scrum, Sakarya Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 2020, **24**(5), 882-891.

[2] **Mercan Ş.**, Durdu P. O., Evaluating the Usability of Unity Game Engine From Developers' Perspective, 2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT), Moscow Russia, 20-22 September 2017.

ÖZGEÇMİŞ

İlk orta ve lise öğrenimimi İstanbul'da tamamladım. 2011 yılında girdiğim Kocaeli Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü'nden 2015 yılında mevzun oldum. 2016 yılında Kocaeli Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümü'nde yüksek lisansa başladım. 2016 – 2018 yılları arasında Mobilist şirketinde çalıştım. 2018 - 2019 yılları arasında Markakod şirketinde çalıştım. 2019'dan itibaren OBSS şirketinde çalışıyorum.