T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

ÇEVİK YAZILIM GELİŞTİRME YAKLAŞIMI: PERAKENDE SEKTÖRÜNE UYARLAMA

Yüksek Lisans Tezi

AYKUT ŞAHİN

İSTANBUL, 2018

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ
MÜHENDİSLİK YÖNETİMİ

ÇEVİK YAZILIM GELİŞTİRME YAKLAŞIMI: PERAKENDE SEKTÖRÜNE UYARLAMA

Yüksek Lisans Tezi

AYKUT ŞAHİN

Tez Danışmanı: DR. ÖĞR. ÜYESİ ADNAN ÇORUM

İSTANBUL, 2018

T.C.
BAHÇEŞEHİR ÜNİVERİTESİ

Tezin Adı: Çevik Yazılım Geliştirme Yaklaşımı: Perakende Sektörüne Uyarlama

Öğrencinin Adı Soyadı: Aykut ŞAHİN
Tez Savunma Tarihi: 25.05.2018

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğu Fen Bilimleri Enstitüsü tarafından onaylanmıştır.

Dr. Ögr. Üyesi Yüce Batu SALMAN
Enstitü Müdürü
İmza

Bu tezin Yüksek Lisans tezi olarak gerekli şartları yerine getirmiş olduğunu onaylarım.

Doç. Dr. Gül Tekin TEMUR
Program Koordinatörü
İmza

Bu Tez tarafımızca okunmuş, nitelik ve içerik açısından bir Yüksek Lisans tezi olarak yeterli görülmüş ve kabul edilmiştir.

Jüri Üyeleri	İmzalar
Tez Danışmanı Dr. Öğr. Üyesi Adnan ÇORUM	…….
Üye Dr. Öğr. Üyesi Betül ERDOĞDU ŞAKAR	…….
Üye Doç. Dr. Özlem ŞENVAR	…….

TEŞEKKÜR

Tez hazırlama sürecinde değerli fikirleriyle beni yönlendiren ve yardımlarını hiçbir zaman esirgemeyen, çok kıymetli hocam ve danışmanım Sayın Dr. Adnan Çorum'a teşekkürü bir borç bilirim. Ayrıca, bu süre zarfında hiçbir fedakârlıktan kaçınmayan ve desteği ile daima yanımda olan değerli eşim Tuğçe Tan Şahin'e teşekkürlerimi sunuyorum.

İstanbul, 2018 Aykut ŞAHİN

ÖZET

ÇEVİK YAZILIM GELİŞTİRME YAKLAŞIMI:
PERAKENDE SEKTÖRÜNE UYARLAMA

Aykut Şahin

Mühendislik Yönetimi

Tez Danışmanı: Dr. Öğr. Üyesi Adnan ÇORUM

Mayıs 2018, 44 sayfa

Global ekonomideki gelişmeler, tüketicilerin demografik yapısındaki değişkenlik ve müşterilerin seçebilecekleri alternatiflerin hızla çoğalması gibi unsurlar perakende sektöründeki kırılganlığı artırarak, yoğun bir rekabet ortamına neden olmaktadır. Bu ortamın yaşandığı şartlarda kendi yerlerini sağlamlaştırmaya ve devamlılıklarını sürdürmeye çalışan perakendeciler, tedarik zinciri yönetimi kavramına daha fazla önem vermeye başlamıştır. Şüphesiz tedarik zinciri yönetiminin tüm unsurlarını bir arada tutmak ve efektif bir biçimde çalıştırmak için güçlü teknolojik altyapılara ihtiyaç duyulmaktadır. Bu teknolojik altyapılar, özellikle hem yazılım geliştirme süreçlerini optimize etmek hem de perakende sektöründeki hızlı değişime adapte olmak için çevik yöntemlere ihtiyaç duymaktadır.

Bu tez çalışması, Türkiye'de faaliyet gösteren ulusal bir perakende şirketinde, tedarik zinciri yönetimi, yazılım projelerinin çevik yazılım geliştirme yaklaşımını kullanarak hayata geçirilmesini amaçlamaktadır. Bu amaca erişilebilmesi için öncelikle bugüne kadar oluşturulmuş çevik çerçeveler incelenmiş ve ihtiyaca göre farklı çevik ilkeler bir araya getirilerek yeni bir çevik çerçeve oluşturulmuştur.

Tez çalışması kapsamında oluşturulan çevik çerçeve tedarik zinciri çözümleri iş birimine ait adres bazlı stok yönetim projesine uygulanarak başarısı ve çıktıları ölçümlenmeye çalışılmıştır.

Anahtar Kelimeler: Çevik Metodoloji, Çevik Çerçeve, Perakende, Tedarik Zinciri Yönetimi

ABSTRACT

AGILE SOFTWARE DEVELOPMENT APPROACH:
ADAPTATION TO THE RETAIL SECTOR

Aykut Şahin

Engineering Management

Thesis Supervisor: Assist. Prof. Adnan ÇORUM

May 2018, 44 pages

Developments in the global economy, consumer demographic variability and rapidly increasing alternatives for customer preference are causing intense competition by increasing the fragility of the retail industry. Retailers, who are trying to consolidate their own space and maintain their continuity, have begun to pay more attention to the concept of supply chain management. There is no doubt that strong technological infrastructure is needed to keep all the elements of supply chain management together and to operate effectively. These technological infrastructures need agile methods to optimize software development processes and adapting rapidly in the retail sector.

This thesis aims to be implemented using the agile software development approach in software project of the supply chain management business unit which retail company operating in Turkey. In order to be able to reach this aim, agile frameworks that were created up to date were examined and a new agile framework was created by combining different agile principles according to their needs.

The agile framework established within the scope of the thesis study was applied to the address-based stock management project of the business unit and the success and output were tried to be measured.

Keywords: Agile Methodology, Agile Framework, Retail, Supply Chain Management

İÇİNDEKİLER

TABLOLAR

ŞEKİLLER

KISALTMALAR

ABS	:	Adres Bazlı Stok
ASD	:	Adaptive Software Development
FDD	:	Feature Driven Development
JAD	:	Joint Application Development
PL/SQL	:	Procedural Language/Structured Query Language
RAD	:	Rapid Application Development
RUP	:	Rational Unified Process
SQL	:	Structured Query Language
TDD	:	Test Driven Development
WIP	:	Limit Work in Progress
XP	:	Extreme Programming

1. GİRİŞ

Günümüzde birçok yazılım projesi, perakende sektörü gibi ihtiyaçları ve pazar koşulları sürekli değişikliğe maruz kalan sektörlerde başarısızlığa uğramaktadır. Bu başarısızlığın temel sebepleri arasında, gereksinimlerin sürekli değişiklik göstermesi, net ve doğru olarak tespit edilememesi, süreçlerin iyi takip edilmemesi ve yanlış proje yönetim metotları kullanılması gösterilebilir. Projelerin başarısızlıkla sonuçlanmasına neden olan bu faktörlerin ortadan kaldırılması için, değişikliğe hızlı adapte olabilen, şeffaf ve doğru proje yönetim yaklaşımı belirlenmesi önem arz etmektedir.

Çevik (Agile) yazılım geliştirme metodolojisi, en kısa sürede değer üreten bir çıktı elde etmeye odaklanmaktadır. Pazar koşullarının sürekli değişiklik gösterdiği, ihtiyaçların tam olarak belirlenemediği karmaşık projeler için oldukça kullanışlı bir yöntemdir.

Bu tez çalışmasının amacı perakende sektör dinamiklerine cevap verebilecek ve yazılım geliştirme süreçlerini optimize hale getirecek bir çevik yazılım geliştirme yaklaşımını ortaya çıkarmaktır. Bu nedenle, günümüze kadar oluşturulmuş çevik çerçeveler (Agile Framework) detaylı olarak araştırılmış ve uygulama örnekleri ile tezin ikinci bölümünde anlatılmıştır. Tezin üçüncü bölümünde ise, mevcut çevik yazılım geliştirme çerçevelerinin ilkelerinden/pratiklerinden, perakende ve tedarik zinciri yönetimi iş yapış şekline uygun pratiklerin seçimi yapılarak, yeni bir çevik çerçeve oluşturulması amaçlanmıştır. Seçimi yapılan ilkelerin faydaları ve takım üzerinde ki etkilerine değinilmiştir. Yeni çevik çerçeveyi ulusal bir perakende şirketinin, tedarik zinciri yönetimi iş birimi altında, adres bazlı stok yönetim projesinde uygulanması ve yapılan işlerin uygulama örnekleri ile anlatımı dördüncü bölümünde yapılmıştır.

Son bölümünde ise, tez kapsamında oluşturulan çevik çerçevenin proje üzerindeki çıktıları, sonuçları ve başarısı ölçümlenerek anlatımı yapılmıştır.

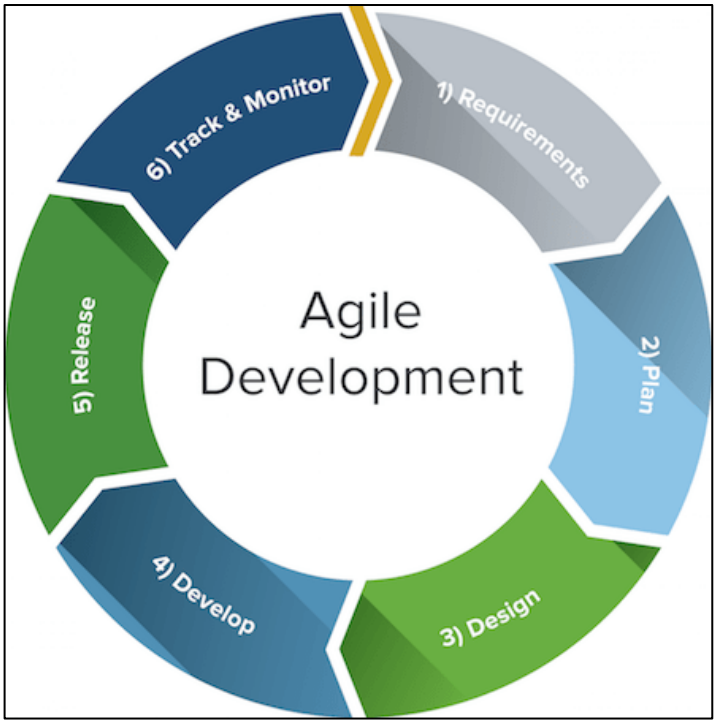
2. ÇEVİK YAZILIM GELİŞTİRME YAKLAŞIMI

Çevik yazılım geliştirme yaklaşımları, üretim alanındaki verimliliğin artırılması amacıyla yalın yaklaşımların, bilgi teknolojileri ve yazılım sektörüne uyarlaması olarak 1950'lerde ortaya çıkmıştır. 1970 yılı itibariyle yazılım sektöründe çeşitli çevik yazılım geliştirme yaklaşımlarına rastlanabilmekle birlikte, çevik yazılım metodolojilerinin kullanımı 1990'ların sonuna doğru hız kazanmış ve son 8 yıl içinde tüm dünyada oldukça başarılı olarak yaygınlaşması arttırılmıştır. Şu anda, sektörde faaliyet gösteren birçok yazılım şirketi ve birçok yazılım geliştirme projelerinde, çevik yaklaşımlar kullanılmaktır. (Beck, vd., 2001).

Çevik yazılım geliştirme yaklaşımı, tekrarlanan yazılım geliştirme metodu baz alınarak geliştirilmiştir, sık aralıklarla parça parça yazılım teslimatını ve değişikliği teşvik eden bir yazılım geliştirme metodolojisidir. Çevik geliştirme, yazılım geliştirme ekibi içerisindeki iletişimin artırılmasını, parçalar halinde yazılım teslimatını, değişimi ve test odaklı yazılım geliştirmesini ve uyumlu bir planlamayı teşvik eder.

Çevik yöntemlerde, yazılım geliştirme süreçleri yinelemeli (iterasyon) ve artırımlı safhalar halinde uygulanır. Bu yinelemeler sonucunda kullanılabilir bir ürün ortaya çıkarılması ve müşteri veya kullanıcıların geri beslemeleri baz alınarak, var ise değişen ihtiyaçlar doğrultusunda geliştirme süreçleri tekrarlanır. Bu yinelemeli süreçler müşterinin tam olarak istediği ürün ortaya çıkana kadar devam ettirilir. Çevik yöntemlerde uygulanan safhalar Şekil 1.1'de gösterilmiştir.

Şekil 1.1: Çevik Yöntem Uygulama Aşamaları



Kaynak: Watts, J., 2017

2.1 ÇEVİK YAZILIM GELİŞTİRME MANİFESTOSU

Dünyanın önde gelen çevik yazılım geliştiricileri (Scrum, Kanban, XP gibi metodolojilerin yaratıcıları) ortak bir zeminde buluşabilmek için 2001 yılında bir araya gelerek "çevik yazılım geliştirme manifestosu" nu ve "çevik yazılım geliştirme prensipleri" ni yayınlamışlardır. Böylelikle çevik metotların projelere genel bakış açıları net bir biçimde ifade edilmiştir.

Bu manifestoda;

i. Bireyler ve aralarındaki etkileşimlerin, kullanılan araç ve süreçlerden;
ii. Çalışan yazılımın, detaylı dokümantasyondan;
iii. Müşteri ile iş birliğinin, sözleşmedeki kesin kurallardan;
iv. Değişikliklere uyum sağlayabilmenin, mevcut planı takip etmekten;

Daha önemli ve öncelikli olduğu belirtilmektedir.

2.2 ÇEVİK YAZILIM PRENSİPLERİ

i. Öncelikli olan müşteri memnuniyetini sağlamaktır. Bu nedenle, öncelikle sürekli kaliteli yazılım teslimatı yapılmalıdır.
ii. Projenin hangi aşamasında olduğunun bir önemi olmadan tüm değişiklikler kabul edilir. Çevik yazılım süreçleri değişiklikleri müşteri avantajına dönüştürürler.
iii. Proje tipine göre mümkün olabilen en kısa zaman aralıklarıyla çalışan, kaliteli yazılım teslimatı yapılır.
iv. Analistler, uzmanlar, yazılımcılar, testçiler vs. tüm ekip elemanları bire bir iletişim halinde ve birlikte çalışırlar.
v. Takım üyelerine gerekli destek verilmeli, ihtiyaçları karşılanarak, güvenilmelidir. Başarılı projeler motivasyonu yüksek bireyler tarafından kurulur.
vi. Doğru bilgi akışı için yüz yüze iletişim önemlidir.
vii. Çalışan yazılım, projenin ilk gelişim ölçütüdür.
viii. Çevik süreçler mümkün olduğunca sabit hızlı, sürdürülebilir geliştirmeye önem verir.
ix. Güçlü teknik alt yapı ve tasarım çevikliği arttırır.
x. Basitlik önemlidir.
xi. En iyi mimariler, gereksinimler ve tasarımlar kendi kendini organize edebilen ekipler tarafından yaratılır.
xii. Düzenli aralıklarla ekipler kendi yöntemlerini gözden geçirerek verimliliği arttırmak için gerekli iyileştirmeleri yaparlar.

2.3 ÇEVİK METODOLOJİLERİN GETİRİLERİ

2.3.1 Düşük Risk

Çevik yazılım geliştirme yaklaşımı, tekrarlanan yazılım geliştirme metotları sayesinde proje risklerini en aza indirgeyip başarıyı arttırmakta ve hata oranlarını düşürerek verimliliği yükseltmektedir. Bunun arkasında yatan en temel etken, projenin başlangıcında projenin temel taşlarının, program parçacıkları halinde geliştirilmesi, proje ekibinin yeteneklerinin ve projede kullanılan tüm araçların önceden tecrübe edilerek eksikliklerinin görülebilmesidir. Ayrıca parçalar halinde geliştirme süreci sayesinde,

proje hızlı olarak şekillenmekte ve proje başlangıcında fark edilemeyen unsurlar, ciddi sorunlara neden olmadan önce öngörülebilir hale gelmektedir.

2.3.2 Değişimin Teşvik Edilmesi

Çevik yazılım geliştirme yaklaşımında belirtildiği gibi, yazılım projelerinde değişiklik olması kaçınılmaz bir durumdur. Orta ve küçük çaplı projelerde dahi proje süresince, proje başlangıcına göre ortalama yüzde 30 oranlarında değişime uğramaktadır. Bu nedenle, değişime uğramak yazılım projelerinin doğasıdır ve bu gerçeklik çevik yaklaşımların üzerinde önemle durduğu bir etkendir. Çevik yazılım geliştirme metodolojileri, değişime karşı gelmek yerine, değişimi müşteri avantajına dönüştürmeye yönelik olarak çalışırlar. Çevik metodolojiler, önerdiği parçalı yazılım üretimi ve her adımdaki güçlü bilgi alışverişiyle, değişim gereksinimlerinin mümkün olduğunca projelerin başlangıç adımlarında fark edilmesini ve projenin değişime hızlı bir şekilde adapte olabilmesini sağlarlar.

2.3.3 Karmaşıklığın Yönetimi

Yazılım projelerinin kapsamları arttıkça, karmaşıklık oranları artmakta ve buna bağlı olarak hata oranları da yükseklik arz etmektedir. Çevik yöntemler ise projeleri, daha küçük ve yönetilebilir parçalara bölerek ele alırlar. Böylece projenin kapsamı ve büyüklüğü ne olursa olsun, küçük parçacıklarla ele alınan yazılım projelerinde karmaşıklık en düşük seviyeye indirilerek, yönetilebilir bir forma dönüştürülür.

2.3.4 Sürekli Yazılım Teslimi

Çevik metodolojilerde her bir yineleme sonunda çalışan bir yazılım parçacığı meydana gelmektedir. Projenin ilk gününden itibaren devam ederek büyüyen ve birleştirilen parçacıklar, müşterilere görebilecekleri ve kullanabilecekleri bir yazılım ürünü sunar ve bu durum da müşteri memnuniyetini arttırmaktadır. Müşterilerin yazılımı tamamlanmış

ancak çalışmayan uygulamalar yerine, çalışır durumda ama tamamlanmış uygulamaları tercih ettiği gözlemlenmektedir.

2.3.5 Yüksek Kalite

Tekrarlanan yazılım geliştirme metotlarının ana fikri test odaklı kaliteli yazılım geliştirmedir. Projenin başlangıcından itibaren tüm test süreçlerinin yazılım geliştirme süreci içerisinde beraber yürütülmesi sayesinde, ortaya çıkabilecek hatalar büyümeden fark edilerek hızlı bir biçimde düzeltilebilmektedir. Proje kapsamı ve hacmi büyüdükçe hata oranları da buna bağlı olarak artış göstermektedir. Ancak çevik yaklaşımlar, projeleri parçalara bölerek ve her bir parçayı kendi gelişimi içerisinde test ederek hata oranlarının düşürülmesini sağlamaktadır.

2.3.6 İhtiyaçlara Daha İyi Cevap Veren Çözümler

Çevik yazılım geliştirme döngüleri/yinelemeleri arasında müşteri ihtiyacını en iyi derecede karşılayabilecek programları oraya koymak için sürekli fikir alışverişi yapılır. Bu noktada müşterinin ihtiyacı ve program hakkında ki düşünceleri ve ihtiyaca karşılık verme oranları sürekli analiz edilmektedir. Müşteriler çoğunlukla proje başlangıcında tam olarak ne istediklerini ortaya koyamazlar, fikirleri yoktur ve istekler, projenin gelişim süreci ile birlikte şekillenmektedir. Ayrıca bilinirliği az olan veya yeni uygulamaya koyulacak bir geliştirme hakkında müşterinin fikir beyan etmesi oldukça zordur. Bu nedenle gerçeklik karşısında, çevik yöntemler müşteri odaklı, hızlı adapte olan ve esnek yapısıyla, müşteri memnuniyetini en üst seviyede tutmayı başarabilmektedir (Larman, 2007, Schwaber, 2003).

2.4 ÇEVİK YAZILIM GELİŞTİRME ÇERÇEVELERİ

Küresel olarak birçok yazılım şirketi çevik olabilmek adına, günümüze kadar oluşturulmuş çerçeveleri kendi yazılım geliştirme süreçlerinde kullanmaya çalışmışlardır.

Aslında bu çerçeveler kullanıldığı proje, mevcut insan kaynağı, teknolojik altyapı ve sektör dinamiklerine göre tercih farklılıkları göstermektedir.

Günümüze kadar oluşturulmuş çevik çerçeveler:

2.4.1 Extreme Programming (XP)

Extreme Programming, gereksinimlerin sıkça ve belirsiz olarak değiştiği şartlarda, yazılım proje takımlarının başarılı ve hatasız yazılım geliştirmesine yardım eden yazılım mühendisliği pratiklerini ifade eder.

i. Sürekli geliştirme,
ii. Sürekli entegrasyon,
iii. Kısa tekrarlamalar,
iv. Küçük sürümler yayınlamak,
v. Hızlı geri bildirim almak,
vi. Müşterinin katılımı ve geri bildirimleri,
vii. Sürekli iletişim ve koordinasyon,
viii. Çift halinde programlama,
ix. Sürekli test

XP'nin temel karakteristikleridir (Abrahamsson vd., 2003; Beck, 1999; Campanelli ve Parreiras, 2015).

XP'de testler yazılım parçacıkları olan bileşenler üzerinde yazılım geliştirme üyeleri tarafından anlık olarak gerçekleştirilir. Tüm yapılan testler bir araya getirilir ve tüm bileşenler bu testlerden geçmek zorundadır. Bir başka tabirle, Test güdümlü geliştirme (Test Driven Development [TDD]) söz konusudur.

Test güdümlü geliştirme önce testler hazırlanır, kodlanır ve üzerinde gerekli düzenlemeler yapılır. Test kapsamı bilinmeden yazılım geliştirmeye geçilmez. Akademik ve endüstriyel çalışmalar incelendiğinde, test güdümlü yazılım geliştirme sürecinde yazılım iç kalitesi, dış kalitesi ve üretkenliğe etkileri incelenmiş ve iç kalitede yüzde 76 ve dış kalitede yüzde 88 avantaj sağladığı rapor edilmiştir (Bissi vd., 2016).

Diğer taraftan müşteriler, her bir tekrarlama için iş odaklı, test edilebilir, sonucu tahmin edilebilir fonksiyonel testler hazırlar ve bileşenler bu testlerin tümünden geçmesi planlanır. Bazı durumlarda müşteri olası geçilemeyen test çözümlemenin zaman gecikmesine neden olacağı ve maliyeti artıracağını değerlendirerek, problemin çözümünden bilerek vazgeçebilir. Müşteri tam zamanlı olarak takımla birlikte çalışır. Tekrarlamalar sonucunda üretilen yeni kod, aktif olarak çalışan canlı sisteme zaman kaybetmeden hızlıca entegre edilir. Entegrasyon esnasında tüm sistem sıfırdan tüm testlere tabi tutulur. Aksi halde değişikliklerin entegrasyonu iptal edilir ve aktif tekrarlamaya geri dönülerek sorun çözümlenmeye çalışılır (Beck, 1999). Her entegrasyon testine sürecin hızlanması ve daha doğru çalışması adına test otomasyonu yazılması tavsiye edilmektedir.

2.4.2 Scrum

Scrum, insanların mümkün olan en yüksek değere sahip ürünleri üretken ve yaratıcı bir şekilde geliştirirken, karmaşık ve adaptasyona açık sorunları ele alabildikleri bir çerçevedir;

i. Basittir
ii. Anlaması kolaydır
iii. Ustaca yönetmek zordur.

Scrum, 1990'ların başından beri karmaşık ürün geliştirme sürecini yönetmek için kullanılan bir çevik çerçevedir. Scrum, bir ürün geliştirme tekniği veya süreci değildir; içerisinde çeşitli süreçleri ve teknikleri kullanabileceğiniz bir çerçevedir. Scrum, ürün yönetimi ve geliştirme pratiklerinizin etkililiğini açık bir şekilde ortaya koyarak iyileştirme fırsatı sunar. Scrum çerçevesi, Scrum Takımları ve takımlarla ilgili rolleri, etkinlikleri, eserleri ve kuralları kapsar. Çerçevedeki her bir bileşen özel bir amaca hizmet eder; Scrum'ın başarısı ve kullanımı için bu zorunludur. Scrum'ın kuralları; etkinlikleri, rolleri ve eserleri birbirine bağlar ve aralarındaki ilişkiler ile etkileşimleri düzenler.

Scrum Teorisi, Scrum'ın temelinde deneysel süreç kontrol teorisi (veya deneycilik) yer alır. Deneycilik, bilginin deneyimden ve bilinen şeylere dayanarak alınan kararlardan

meydana geldiğini ileri sürer. Scrum, öngörülebilirliği en iyi seviyeye çıkarmak ve riski kontrol etmek için iterasyonlu ve artımlı (incremental) bir yaklaşım kullanır. Her deneysel süreç kontrol uygulaması üç ayakla desteklenir: şeffaflık, gözlem ve adaptasyon.

Şeffaflık; çıktıdan sorumlu kişiler sürecin önemli kısımlarını izleyebilmelidir. Şeffaflık bu kısımların bir ortak standartla tanımlanmasını gerektirir. Bu sayede bakan kişiler gördüklerinden aynı şeyi anlarlar.

Örnek olarak:

i. Tüm katılımcılarla sürece ait ortak bir dil paylaşılmalı;
ii. İşi yapanlar ve işin sonucunu bekleyenler ortak bir "Bitti" (Done) tanımına sahip olmalıdır.

Gözlem; Scrum uygulayanlar, istenmeyen sapmaları tespit edebilmek için Scrum eserlerini ve Sprint Hedefine doğru ilerlemeyi sıkça gözlemlemelidir. Bu gözlemler, iş yapmaya engel olacak kadar sık olmamalıdır. Gözlemler, çalışma esnasında yetkin gözlemciler tarafından itinayla yapıldığında en çok faydayı sağlar.

Adaptasyon; şayet bir gözlemci sürecin bir veya daha fazla kısmının kabul edilebilir sınırlar dışına çıktığını ve sürecin sonunda çıkacak ürünün kabul edilemez olacağını tespit ederse üzerinde çalışılan süreç veya ürün düzeltilmelidir. Düzeltme daha fazla sapmaya izin vermeden mümkün olan en yakın zamanda yapılmalıdır.

Scrum, gözlem ve adaptasyon için bu Scrum Etkinlikleri olarak tarif edilen dört resmî etkinliği (toplantıyı) zorunlu kılar:

i. Sprint Planlama (Sprint Planning)
ii. Günlük Scrum (Daily Scrum)
iii. Sprint Değerlendirme (Sprint Review)
iv. Sprint Retrospektifi (Sprint Retrospective)

(Scrum Tanımlayıcı Kılavuzu, Scrum Guide, 2003)

2.4.3 Yalın Yazılım Geliştirme

Yalın Yazılım Geliştirme olgusu ilk kez Avrupa Birliği'nin ESPRIT girişimi tarafından Almanya'nın Stuttgart şehrinde Ekim 1992'de organize edilen konferansın başlığı olarak kullanılmıştır. Bağımsız olarak, 1993 yılında Robert "Bob" Charette yazılım projelerinde riskleri daha iyi risk yönetimini araştıran çalışmasının bir parçası olarak "Lean Yazılım Geliştirme" kavramını önermiştir. "Yalın" terimi James Womack, Daniel Jones ve Daniel Roos tarafından 1991 yılında "The Machine That Changed The World: The Story of Lean Production" (Dünyayı Değiştiren Makine: Yalın Üretim Hikayesi, 2017) kitabında Toyota'da kullanılan yönetim yaklaşımını ifade etmek üzere kullanılan İngilizce terim olarak önerilmiştir. Yalın düşüncesinin yazılım geliştirmede uygulanabilir olacağı fikri çok önceleri, bu terim üretim süreçleri ve endüstriyel mühendislik alanındaki trendlerle ilişkili olarak ilk kez kullanıldıktan 1 ve 2 yıl sonra yerleşmiştir.

Womack ve Jones'un 1995'de yayınladıkları 2. Kitaplarında (Womack ve Jones, 2003) Yalın düşünmenin beş ana öğesi tanımlanmıştır. Bunlar:

i. Değer
ii. Değer Akışı
iii. Akış
iv. Çek
v. Mükemmellik

Bu ilerleyen on yıldan daha fazla süre boyunca Yalın için varsayılan tanım haline geldi. Önerildiği üzere mükemmellik hedefine fazlalığı yok ederek ulaşılmıştır. Beş dayanak olmasına karşın, bu beşinci olanıydı, zararlı etkinliklerin ve bunların ortadan kaldırılmasının sistematik tanımlamasıyla mükemmellik arayışı geniş bir kitleye yankılanmıştır. Yalın, 1990'ların sonlarında ve 21. Yüzyılın başında özellikle önleme uygulamalarıyla ilişkilendirilmiştir.

Womack ve Jones'un Yalın tanımı evrensel olarak kabul edilemez. Toyota'daki yönetim prensipleri çok inceliklidir. Türkçe 'deki "Atık" kelimesi üç Japonca terimle daha zengin bir şekilde açıklanmaktadır:

i. Muda – gerçek anlamda “atık” demektir, ancak katma değeri olmayan etkinliği belirtir.
ii. Mura- “düz olmayan” anlamına gelmektedir ve akışta çeşitlilik olarak yorumlanır
iii. Muri- “fazla yüklenme” veya “mantıksızlık” anlamına gelmektedir.

Bu bilgiler ışığında, yalın yazılım geliştirmenin en öncelikle felsefesinin üretim sürecindeki atıkları atmak olduğu söylenebilir. Atık, kısmi yapılmış işler, ekstra yapılan işlemler, program hataları, ekstra özellikler, görev değişiklikleri, beklemeler, hareketlilik ve program kusurları gibi müşteri açısından ürünün değerine katkısı olmayan her şeydir. (Poppendieck ve Poppendieck 2013). Bu şekilde sadece müşterinin istediğini karşılayacak ürünü elde etme süreci yürütülmüş olur. Süreç boyunca tüm varsayımlar tekrar tekrar doğrulanır. Bir metrik veya pratik artık geçerli değilse çöpe atılır. Kısa yinelemeler sonunca geri beslemeler alarak ürün doğrulama (validation) gerçekleştirilir. Kararlar, mümkün olduğunca ertelenir ve alınması gereken son zamanda alınır. Bu şekilde kesinleşmemiş kararlara uygun atık işlemlerle gereksiz yere uğraşılmamış olunur.

2.4.4 Kanban

Kanban, üretim operasyonlarından türetilmiş, adaptasyonu yüksek, görsel ve maliyet odaklı bir tekniktir. 1950’lerde Toyota tarafından kullanılmasına rağmen, yazılım geliştirme sürecine uygulanması ilk kez, 2004 yılında David J. Anderson tarafından gerçekleştirilmiştir. David J. Anderson, Microsoft firmasında bir takımla birlikte çalışırken problemlerin giderilmesi amacıyla bu yöntemi ortaya çıkarmıştır (Ahmad vd., 2013).

Kanban arkasındaki temek fikir yalın düşünceden gelmektedir. Kanban yönteminin ana felsefeleri üç bölümde değerlendirilebilir; iş akışını görselleştirmek, üzerinde çalışılan işleri sınırlamak (Limit Work In Progress [WIP]) ve döngü sürelerini ayarlamak. Kanban yönteminde yapılan ve olası yapılacak işler “Kanban Board” denilen bir tahtada görselleştirilir. Bu sayede tüm takım ve müşteri; mevcut işerlerin son durumunu, öncelliklerini, olası darboğazları görerek aksiyon alır. Bu tahtada soldan sağa; yapılacak işler (ToDo), yapılmakta olan işler (In-Progress) ve testi geçen ve biten işler (Done) not

kağıtlarına yazışmış şekilde bulunmaktadır. Süreç boyunca bu not kağıtları soldan sağa doğru hareket etmektedir. Amaç başlangıçta yapılacak işler listesindeki işi, biten işler bölümüne kadar hareket ettirmektir. Bununla birlikte önemli unsurlarda biri de WIP'i mümkün olduğunda küçük tutmaktır. Bu sayede yazılım geliştiren personel verilen bir zaman aralığında sadece bu işe odaklanır ve müşteri de o zaman aralığı sonunda ne elde edeceğini bilir. Zaman baskısı yönünden incelendiğinde Scrum'dan farklı olduğunu söylenebilir, nitekim kısa geri besleme döngüleri ile hızlı adapte olmayı hedefler. Kanban kullanmanın anahtar güdüsü zorlayıcı yinelemeler değil, akışa odaklanmaktır (Ahmad vd., 2013).

Kanban metodunda uygulama üzerinde ki kusurlar bir atık olarak görüldüğünden dolayı, bu unsurların yok edilmesi, yok edilemiyorsa şayet minimum hale getirilmesi hedeflenmektedir. Uygulama kusurundan kaynaklanan atığın miktarı, kusurun ürüne etkisi ve fark edilmeme süresi olarak görülür. Örneğin, üç dakika içerisinde fark edilmiş bir kritik kusur büyük bir atık kaynağı olarak sayılmaz. Haftalarca fark edilememiş küçük bir kusur ise daha büyük bir atık olarak sayılır. Kusurların etkilerini azaltmanın yolu, kusurun oluşur oluşmaz onları tespit etmektir. Böylece harcanan zamanı azaltmak için kod yazarken anında test, sık sık entegrasyon ve mümkün olabildiğince çabuk sürüm yayınlanması yapılır (Poppendieck M. ve Poppendieck T., 2013).

Ayrıca, bu bilgilere ek olarak son zamanlarda Scrum ve Kanban bazı özellikleri bir araya getirilerek "Scrumban" isimli bir hibrit çevik yöntem çalışmalarına rastlanmaktadır. Bu yöntemde Kanban yönteminde olduğu gibi işler yapılacaklar tahtasına alınır ama farklı olarak bazı işler için zaman sınırı olduğunu gösterir farklı renkte kâğıtlar kullanılır (Reddy, 2014).

2.4.5 Feature Driven Development (FDD)

FDD, yazılım geliştirme aşamalarında sadece tasarım ve geliştirme süreçlerini etkileyen çevik ve adaptasyonu yüksek bir yazılım geliştirme yöntemidir. Ana aktiviteleri beş bölümde özetlenebilir:

i. Tüm sistemi modelleme,

ii. Özellik listesi hazırlama,
iii. Özellikleri planlama,
iv. Özelliğe göre tasarım,
v. Özelliğe göre geliştirme

Sistemin modelleme safhasında, takım üyeleri ve varsa danışmanlar sistem için bir "yol planı" oluştururlar. İkinci safhada, yazılım ürününde ihtiyaç olan özellikler tüm takım üyeleri ve uzmanların anlayabileceği basitlikte parçalara ayrılır. Üçüncü ve son aşamada "tasarım paketleri" adı verilen parçalar önceliklendirilir ve her bir parça, programcılardan sorumlu bir şef programcıya atanır. Yazılım parçacığının özelliğine göre tasarım aşamasında sürecin yinelemeli kısmı başlar. Şef programcı kendisine atanmış tasarım paketinden 1-2 hafta içerisinde tamamlayabileceği bir alt iş seçer. Geliştirme aşamasında bu alt iş yeniden ayrıntıları ile analiz edilir, tasarlanır, kodlanır, test edilir ve entegre edilir (Palmer & Felsing, 2001). FDD, diğer çevik yöntemlerle kıyaslandığında, büyük ve kritik projelerde kullanılmasının oldukça uygun olduğu söylenebilir nitekim tüm yazılım geliştirme süresi boyunca kaliteye ağırlık verilir ve kaliteden hiçbir müddetçe taviz verilmez.

FDD yönteminde, yazılım geliştirmeler tamamlandıktan sonra birim testi (Unit Test) ve kod inceleme (Code Review) safhalarına geçilir. Hangi testin önce yapılacağı şef programcı tarafından karar verilir. Birim testin, manuel mi veya otomasyon aracılığı ile yapılacağına karar verilir. Kod inceleme FDD için zorunlu bir kuraldır ve bunun için yazılım geliştirme süresinin dörtte biri kadar zaman ayrılır. Kod inceleme, iyi bir biçimde yapıldığında, kod içinde ki yazılım kusurları tespit edilebilmekte ve ilerde yaşanabilecek karmaşıklıklarının önüne geçilebilmektedir (McConnell, 2004). Bu inceleme safhası sadece yazılım geliştiriciler tarafından yapılmakla kalmayıp, takım içindeki tecrübeli programcılar tarafından da bilahare yapılmaktadır. Bu durum ekibe yeni katılmış az tecrübeli programcıların gelişmesine de yardımcı olmaktadır. Ayrıca proje kapsamında yazılan kodların başka bir programcı tarafından incelendiğini bilmek, daha dikkatli ve kod standartlarına sadık kalarak uygun kod yazılmasını da sağlar. Kod incelemenin seviyesi, yinelemede geliştirilen özelliğin etkisi ve karmaşıklığına göre şef programcı tarafından belirlenir. Önem ve karmaşıklık derecesine bağlı olarak daha az öneme sahip işler takım tarafından incelenmesi yeterli iken, daha karmaşık ve önemli işler, şef

programcılar ve bazen de diğer takımlardan programcılar ile incelenmektedir. Bu sayede XP programlama da yapılan iki kişi programlama pratiğinden daha iyi sonuçlar elde edilir. Bu noktada programcıya koda bir süre sonra tekrardan göz atma şansı verilmekle birlikte, kodlara farklı insanların bakması ve fikir yürütmesi ile çok daha kaliteli, hızlı ve doğru kodların elde edilmesi mümkün olabilmektedir. Tüm bu aşamalar maruz kalan kod, entegrasyona hazır hale gelmektedir (Flora ve Chande, 2014).

2.4.6 Dynamic Systems Development Method (DSDM)

Dynamic Systems Development Method, 1995 yılında sektörde önde gelen proje uzmanları tarafından Hızlı Uygulama Geliştirme (Rapid Application Development [RAD]) için kaliteyi hedeflemek amacıyla geliştirilen ve 1995'de Stapleton ve Millington tarafından tanımlanan bir çerçevedir (Flora ve Change, 2014)

İlk çevik yöntem olarak ortaya çıkan bu çerçeve, proje ürün yönetimi yaşam döngüsünün en başarılı süreçlerini bir araya getiren yinelemeli ve artırımlı bir metodolojidir. Sonuçlara efektif ve hızlı bir biçimde ulaşmayı amaçlayarak, maliyet, risk, zaman ve kaliteyi kontrol altında tutup iş faydalarını artırımsal olarak ulaşmayı sağlamak bu yöntem için stratejik bir amaçtır. Takımın sürekli güçlendirilmesi, kaliteyi sürekli iyileştirmek yerine sürümlerin hızlı ulaştırılmasını sağlamak, geliştirme esnasında herhangi bir değişikliği anında geri alınabilmesi, en üst seviye ihtiyaçlarının değişmemesi, iyi haberleşme ve tüm paydaşlarının birlikteliği ile tüm proje süreci boyunca sürekli test yapılması bu çerçevenin temel prensipleridir. Bu yöntemde kullanılan temel teknikleri arasında, Zaman kutuları, Prototipleme, Test, İmalat ve Modelleme gösterilebilir. Bu çerçeve kapsamında ön-proje, fizibilite çalışması, iş çalışması, fonksiyonel model yineleme, tasarım ve geliştirme yinelemesi, uygulama ve proje sonu aşamaları gerçekleştirilir (Millington ve Stapleton, 1995; Cohen vd., 2004).

Fonksiyonel model tekrarlaması, işin doğru yapıldığını ispat etmek için dokümanlar gözden geçirilip, prototipler sergilenerek yazılım parçasının başarılı bir şekilde test edildiği kanıtlanır. Bu noktada ürünün fonksiyonelliğe katkısı üzerinde durulur. Fonksiyon olmayan kısımlar ise tasarım ve gerçekleştirme aşamalarında test edilir. Kabul

testlerinin zaman kaybetmeden hızlıca geçilebilmesi için kullanıcıların aktif katılımı gereklidir. DSMS test araçlarının kullanımını tavsiye etmektedir. Yakalama ve yeniden oynatma (capture and replay) araçları ile testlerin yapılması, test doğruluklarının kanıtlanması içinde başarılı bir yol olmakla birlikte, kâğıt üzerinden takibinden daha az efor harcayan bir işlem olarak kabul edilir. Kod gözden geçirme (Code Review) DSDM'de önerilen bir pratiktir. Dinamik analiz araçları, demo sırasında dizi boyutları, hafıza kullanımı gibi hususları test etmek için kullanılabilmektedir (Millington ve Stapleton, 1995).

2.4.7 Adaptive Software Development (ASD)

ASD, çok hızlı ve değişken bir yapıda olan internet ekonomisi için geliştirilmiş yazılım geliştirme yöntemidir. Çok hızlı ve değişimi fazla olan projeler, tahmin edilmesi zor ve geleneksel yazılım geliştirme yöntemleri ile yönetilemeyecek kadar karmaşık bir yapıya sahiptir. Bu nedenle yazılım geliştirme yapısının adaptasyonlu olması, sürekli değişime maruz kalan bölümlerde hızlı bir biçimde cevap vermesi gerekir. ASD sürekli prototip geliştirmesi yaparak yinelemeli ve artırımsal geliştirmeyi teşvik eder. Yöntemde "kaosun sınırında dengeleme" mantığı ile, "yeterli kılavuzluk ile projelerin kaosa düşmesi engellenebilir" düşüncesi temeldir. Bu yöntemde proje üç safhada değerlendirilir:

i. Tahmin etmek
ii. İş birliği yapmak
iii. Öğrenmek

Belirsizliklerin az olduğu durumlarda plan yapmak yerine, bu yöntemde tahmin yapılması tercih edilir.

Birlikte çalışma hızla değişen sistem geliştirmede oldukça önemlidir. Hatalar üzerinden bilgi üretme ve geliştirme sırasındaki ihtiyaç değişimlerini sağlamak öğrenme fazı olarak tanımlanmaktadır. Proje başlama, adaptasyonlu çevrim planlama, eşzamanlı özellik geliştirme, kalite gözden geçirme ve final kalite güvence-sürüm yayınlama olmak üzere beş genel adımdan oluşur. Kalite gözden geçirme ile çevrim planlama arasında öğrenme geri beslemesi bulunmaktadır (Highsmith, 2000).

Müşterinin (iş sahibi) geliştirme aşamasında takımla birlikte olması nedeniyle Ortak Uygulama Geliştirme ((Joint Application Development [JAD]) temel olarak alınmaktadır. Müşteri ile birlikte çalışma kabul test sürecinin daha doğru ve kolay geçilmesini sağlar. Bilinirliği az olan veya ilk kez yapılacak işler ve araştırma geliştirme çalışmaları için bu yöntemdeki öğrenme geri beslemesi oldukça fazla önem arz etmektedir. Bu geri besleme ile birlikte kabul testi gerçekleştirilmiş olup hem müşteri bakış açısından hem de teknik açıdan incelemeler yapılmış olur. Bu incelemeler ile birlikte takım kendi performansını değerlendirme şansı bulur. Ayrıca JAD toplantılarında tüm paydaşlarının projenin son durumunu anlık olarak gözden geçirme fırsatı bulur (Abrahamsson vd., 2002; Highsmith, 2000).

2.4.8 Microsoft Solution Framework for Agile (MSF)

Microsoft tarafından 1993 yılında Microsoft Solution Framework (MSF) 'in ilk versiyonu yayınlanmıştır. Sonraki yıllarda; 1997'de versiyon 2.0, 1999'da versiyon 2.5, 2002'de versiyon 3.0 ve en son 2005 yılında ise versiyon 4.0 yayınlanması izlemiştir. (Keeton, vd., 2006).

MSF başarılı bir bilgi teknolojileri süreci için yazılım geliştirme ekibinin nasıl organize edileceği, projelerin nasıl planlaması gerektiği, süreç yapısının nasıl oluşturulacağı, risklerin değerlendirip kurulumlarının nasıl yapılacağı ile ilgili bilgiler veren bir süreç yaklaşımıdır. MSF, süreç ve takım olmak üzere iki farklı modelden oluşmaktadır (Turner M., 2006).

Aşağıdaki prensipleri benimsemektedir:

i. Müşteri ile birlikte geliştirme
ii. Açık iletişim kurmak
iii. Ortak bir vizyonda hareket etmek
iv. Kalitenin herkese ait ortak bir değer olduğuna inanmak
v. Çevik olmak
vi. Günlük kurulum alışkanlığı edinmek

vii. Sürekli gelişim sağlamak

viii. Risk Yönetimi

2.4.9 Rational Unified Process (RUP)

Rational Unified Process, 30 yıllık çalışmalar sonucunda ortaya çıkmıştır. RUP mimariye dayalı ve vaka tabanlıdır, bir başka deyişle "Tekrara dayanan" ve "Artırımsal" bir modeldir. Bu özellikle büyük ölçüde "Objectory" yaklaşımından alınmıştır.

Rational Software firması, RUP'un eksik olan yanlarının tamamlanması amacıyla, bu eksik yanlar konusunda deneyimli olan başka şirketleri ya satın almış ya da şirketlere partnerlik anlaşmaları yapmıştır.

Buna göre, Requisite Inc., Gereksinim Yönetim konusunda, Pure-Atria Konfigürasyonu Yönetimi konusunda, Başarım ve Yük testi konularında katkıda bulunmuşlardır. Bunun yanında geliştirilecek yazılım sisteminin iş süreçlerinin modellenmesi, vaka tabanlı kullanıcı ara yüzü geliştirme alanlarında da yenilikler geliştirilmiştir.

Agile Unified Process ise RUP prensiplerine sadık kalarak, basitlik, çeviklik, geliştirme araçları bağımsız, yüksek değerleri faaliyetlere odaklanma konuları üzerinde yoğunlaşmıştır.

3. PERAKENDE SEKTÖRÜNE UYGUN ÇEVİK YAZILIM GELİŞTİRME ÇERÇEVESİNİN OLUŞTURULMASI

Perakende sektör dinamiklerine ve tedarik zinciri yönetimi projelerine uygun bir çevik çerçeve oluşturulması amaçlanmaktadır. İkinci bölümde araştırılması yapılan çevik yaklaşımlar içinden, proje kapsamı, şirketin mevcut insan kaynağı, proje bütçesi ve teknolojik altyapısına uygun patrikler ve ilkeler seçilmiştir.

Yeni oluşturulan çerçeveye ait pratikler belirlenirken uygulanabilir, şeffaf, adaptasyonu yüksek ve sektör dinamiklerine uygunluğu ön planda tutulmuştur.

Aşağıda yeni çevik çerçeve için belirlenen ilkeler(pratikler) detaylı olarak anlatılmaktadır.

3.1 ÇALIŞAN YAZILIM

Yazılım projelerindeki temel problem, ürünün geliştirmeye başlamasından itibaren çalışan bir yazılımın ortaya çıkmamasıdır. Bu durumda kaliteli bir yazılım ürünü oluşturulmaya çalışılsa dahi, çalışan bir ürün ortaya çıkmadıkça, ortaya koyulan efor da bir anlam teşkil etmeyecektir. Bu durumda çevik metodolojilerin ortak özelliği çalışan yazılım ortaya koyma prensibi ilk prensibimiz olarak karşımıza çıkmaktadır.

Takım üyelerine, çalışan yazılımın önceliğimiz olduğu anlatılmış ve proje süresi boyunca yazılım ihtiyacının temel özelliklerine öncelik verip, her sprint sonunda çalışan bir yazılım ürünü teslim etme sözü verilmiştir.

3.2 MÜŞTERİ MEMNUNİYETİ

Geliştirilecek yazılım ürünü bir müşteri ihtiyacından ortaya çıktığından dolayı, müşterinin oraya çıkacak üründen veya ürün parçacığından mutlaka memnun olması gerekir. Çevik metodolojilerin en önemli temel taşlarından biridir. Müşteri, yinelemelerle

ilerlerken hem ortaya çıkan ürünü görme hem de geleceği planlama ve değişiklik yapma yeteneğinin olması gerekmektedir.
Takım üyelerine yapılacak tüm geliştirmelerde, müşterinin (iş birimi) sprint veya proje sonunda dahi ortaya çıkaracağı bir değişikliği olumlu karşılayıp, bu değişikliği yazılım ürününe entegre etme yönünde iyi niyet gösterilmesi gerektiği anlatılmıştır, çünkü müşteri memnuniyeti bu yöntemde esastır.

3.3 SPRINT (YİNELEMELİ SÜREÇ) İLE GELİŞTİRME

Tüm çevik metotların ortak özeliğidir. Sprintler 1-4 hafta arasında gerçekleştirilir (Scrum Guide, 2003). Her sprint süreci sonunda çalışan bir yazılım ürünü teslim edilir. Takım 2 haftalık sprintlerin kendilerine uygun olduğunu düşünmüştür nitekim 1 haftalık sprint süresi oldukça kısa, bu nedenle adaptasyonu düşük. 3 ve 4 haftalık sprintler ise klasik geliştirme anlayışlarını ortaya çıkaracağını ve zamanı yararlı kullanamayacakları düşüncesi ile 2 hafta olarak belirlenmiştir.

3.4 MÜŞTERİ İLE BİRLİKTE GELİŞTİRME

Yöntem XP yaklaşımından alınmıştır. Müşteri (Product Owner) her yinelemede kendi zamanının yüzde 20'sini geliştirme takımı ile birlikte harcamak durumundadır. Yazılım çözümünün istenilen düzeyde olabilmesi için muhakkak Product Owner ile birlikte geri bildirimler alarak ilerlenmesi gerekmektedir. Ancak bu durumda istenilen ürün tam olarak ortaya çıkabilmektedir.

Bu noktada Product Owner rolünde, iş biriminde ki çalışana koçluk yapılmış ve takım ile daha fazla zaman geçirmesi gerektiği bildirilmiştir. Proje de eksik bilgisi olduğu noktalarda, işi yerinde görme ve gözlemleme yaparak eksik bilgisini gidermesi için gerekli destek verilmiştir.

3.5 OPTİMUM TAKIM PERFORMANSI İÇİN KİŞİ SAYISI

Scrum çerçevesinden alınmıştır. Takım performansının optimum seviyede olabilmesi için oluşturulacak takımın 3-9 kişi arasında olması gerekmektedir. Çünkü 3 kişiden az olması takım oluşumunu engellediği, 9 kişiden fazla olması ise karmaşıklığı artıracağı düşünülmektedir (Schwaber ve Jutherlang, 2003).

Bu bilgiler ışığında takım 4 kişi geliştirme takım üyesi bir kişi de iş biriminden olmak üzere 5 kişiden oluşmuştur. Proje süresi boyunca takımdan 1 geliştirme takım üyesi işten ayrılmış yerine 1 kişi ilave edilmiştir. Yeni katılan takım üyesine gerekli eğitimler verilmiştir. Yöntemin gereklilikleri anlaşılabilir ve uygulanması açık olduğundan dolayı adaptasyon süresi oldukça kısa sürmüştür.

3.6 GÜNLÜK AYAKTA TOPLANTILAR

Scrum çerçevesinden alınmıştır. Günlük toplantılar, 15 dakikalık zaman sınırlı bir aktivitedir. Amacı, geliştirme takımı üyelerinin birbirleri ile senkronize olmalarını sağlamak ve sonraki 24 saat için plan yapmaktır. Bir önceki toplantıdan beri yapılan çalışmaların denetlenmesi ve bir sonraki toplantıdan önce yapılacak çalışmaların tahmin edilmesi ile toplantının amacına ulaşılır. Günlük toplantıları, karmaşıklığı azaltmak için her gün aynı yer ve saatte düzenlenir. Toplantı esnasında, her Geliştirme Takımı üyesi aşağıdaki konular hakkında konuşur:

i. Bir önceki toplantıdan sonra Geliştirme Takımı'nın Sprint Hedef'ine ulaşması için hangi işleri tamamladım?
ii. Bir sonraki toplantıya kadar Geliştirme Takımı'nın Sprint Hedef'ine ulaşması için hangi işleri yapacağım?
iii. Benim veya Geliştirme Takımı'nın Sprint Hedefine ulaşmasını engelleyebilecek bir engel görüyor muyum?

3.7 KALİTE KRİTERLERİNİN BELİRLENMESİ

Kalite kriterlerinin belirlenmesi aşamasında geliştirme takımı kaliteli bir yazılım ürününün hangi adımlardan geçerek oluşabileceğinin belirler.
Takımın belirlediği kaliteli kod adımları:

i. Analiz ve analiz dokümanının oluşturulması
ii. Geliştirme
iii. Kod gözden geçirme
iv. Birim Testi
v. Fonksiyonel Test
vi. Kullanıcı kabul testi
vii. Kullanıcı dokümanı
viii. Teknik doküman
ix. Yaygınlaştırma notu

3.8 KODLAMA STANDARTLARI

XP çerçevesinden alınmıştır. Takım üyeleri kuruma ait tanımlanmış kodlama standartlarına bağlı kalarak yazılım geliştirirler. Kodlama standartlarına bağlı kalarak geliştirilen yazılım karmaşıklığı ortadan kaldırır, herkes tarafından anlaşılır ve sonrasında bakımını kolaylaştırmak amaçlanmıştır.

Kodlama standartları ile ilgili dokümanlar tüm takım üyelerine dağıtılmış ve bu standartlara göre geliştirme yapılması istenmiştir. Zaman zaman kodlama standartları dışında çıkılsa da özellikle kodu gözden geçirme (code review) aşamasında tespit edilmiş ve düzenlemeler anında yapılmıştır.

3.9 PLANLAMA TOPLANTISI VE PLANLAMA OYUNU

Planlama toplantısı Scrum, planlama oyunu ise XP çerçevesinden alınmıştır. Product Owner, her sprint başlangıcında planlama toplantılarına iş listesi ile birlikte gelir. İşin

gerekliliklerini ve niteliklerini ortaya koyar, takım üyeleri ise her bir işe planlama oyunu yaparak bir büyüklük belirler bu büyüklüklere göre zaman içerisinde takımın bir sprint boyunca yapabileceği maksimum iş kapasitesi oluşturulur.

Planlama toplantılarına ilk başlandığında takım mevcut kapasitesinin üstünde iş aldığından dolayı ilk sprint oldukça stresli geçmiş ve yüzde 85 tamamlanma oranına ulaşmıştır. Sonraki sprintlerde bu oran yükselme göstermiş ve tam kapasite tamamlama oranı ile proje sonuna kadar devam etmiştir.

3.10 AYNI ANDA YAPILACAK İŞ SAYISI KISITLAMASI

Bu pratik Kanban çerçevesinden alınmıştır. Bir takım üyesinin aynı anda ilgilenebileceği maksimum iş sayısı vardır. Bu nedenle aynı anda yapılabilecek iş sayısında kısıtlamaya gitme şu anlama gelmektedir; Sprint başında yapılacak işler listesinden bir geliştirme takım üyesi aynı anda bir iş tanımı (user story) ait toplamda maksimum 3 görev seçebilmektedir. Görev eksiltilmeden yeni görev çekmemesi gerekmektedir. Bu sayede hem geliştirme takım üyesi hem de takım kendi kapasitesini yönetebilmesi hem de izlenebilirliğin maksimum düzeye getirilmesi amaçlanmıştır.

3.11 TAKIMIN SPRINT ODAKLI OLMASI

Pratik, Scrum çerçevesinden alınmıştır. Takım üyeleri sadece Sprint içindeki işler ile ilgilenir, bu işler haricinde farklı bir iş ile uğraşmaz. Bu sayede mevcut işler üzerindeki konsantrasyonu yüksek olur ve bu konsantrasyon yazılımdaki kaliteyi en üst düzeye getirir.

Takımın sprint odaklı çalışma prensibi iş birimine detaylı bir biçimde aktarılmıştır. Kendileri de başarılı bir sonuç elde etmek için operasyonel işleri operasyon birimi ile çözmeye çalışmış ve takıma yansıtmamaya çalışmışlardır. Çalışmanın ilerleyen aşamalarında operasyonel iş yüklerinin artması sebebiyle sprint içine toplam kapasitenin yüzde 10 ‘unu geçememek şartıyla operasyonel işler adı altında yeni bir görev tanımlanmıştır.

3.12 BİR MASA ETRAFINDA ÇALIŞMA

Bir masa etrafında çalışma Scrum çerçevesinden alınmıştır, her ne kadar Scrum kılavuzunda bu konuya yer verilmese de uygulama aşamasında tavsiye edilmektedir.
Bir masa etrafında oturma takım arasında etkileşimin ve oluşabilecek dar boğazlarına önüne geçilmesi amaçlanmıştır.

Özellikle yazılım geliştirme takım üyeleri, zaman zaman bireysel çalışmalarında takılı kaldıkları durumlar yaşanabilmektedir. Bazen takılı kaldıkları bölüm çözülmesi güç bir problem olsa da çoğu zaman oldukça basit ve sadece gözden kaçan küçük detaylardan oluşmaktadır. Bu nedenle bir masa etrafında oturma herhangi bir sorunuz ve sorununuz olduğunda diğer takım üyesine danışmanız ile çözülebilmektedir.

Proje süresi boyunca takım üyelerinin bir masa etrafında yüz yüze çalışıyor olması takım performansını en optimum düzeye getirmekle kalmamış aynı zaman da şeffaflığı arttırarak takıma güven empoze etmiştir.

3.13 ORTAK BİR DİL VE AÇIK İLETİŞİM

Bu yaklaşım MSF çerçevesinden alınmıştır. Projenin başarıya ulaşabilmesi için takım üyelerin arasında açık, samimi ve anlaşılabilir ortak bir dil kullanılması ve iletişimin en üst seviyede olması gerekmektedir.

Takımın gerçekleştirdiği tüm toplantılarda aralarında sürekli şeffaflığı ön plana çıkarılması hedeflenmiştir ve bu hedefe ulaşılmıştır. Zaman ilerledikçe takım üyeleri arasında ki bu şeffaf ortam iyi bir arkadaşlık ortamının oluşmasına sebep olmuştur.

3.14 ORTAK VİZYON VE SPRINT AMACININ BELİRLENMESİ

Ortak vizyon yaklaşımı MSF, sprint amacının belirlenmesi ise Scrum çerçevesinden alınmıştır. Ortak vizyon, bir sorunu çözmek için yapılması gerekenleri ortaya koymaktadır.

Ortak vizyona hizmet eden en iyi pratik, sprint başlarında o sprinte sonunda ulaşılması amaçlanan hedefin açık bir biçimde ortaya koyulması ile oluşmaktadır. Bu nedenle takım, sprint sonunda ulaşılması gereken hedefi Product Owner ile birlikte ortaya koymaktadır. Bu pratik takım üyelerini işleri zamanında bitirme ve kaliteli bir yazılım ürünü ortaya çıkarma yolunda kamçılayan bir unsur olmaktadır.

3.15 TEST GÜDÜMLÜ YAZILIM

Test güdümlü yazılım pratiği XP çerçevesinden alınmıştır. Yazılım geliştirmesine başlamadan önce, test senaryoları ve test kodları oluşturulur. Bu pratikteki amaç yazılım kalitesini yüksek tutmaktır. Yazılım çözümü yaygınlaştırılmadan önce mutlaka detaylı test mekanizmalarına tabi olması çalışma süresi boyunca test senaryoları ve kodlamalarına azami dikkat gösterildiğinden dolayı hata oranları oldukça düşük sevilerde seyretmiştir.

3.16 TAM ZAMANINDA TESLİM

Tam zamanında teslim prensibi Scrum çerçevesinden alınmıştır. Takım her 2 haftalık sprintler içerisinde oluşturdukları uygulamaları teslim eder. Bu prensibin, Scrum içerisinde uygulanmasında sıklıkla yapılan bir hata vardır, bu da uygulamaların sprint içerisinde tamamlanmasına rağmen tesliminin sprint sonuna kadar beklenmesidir. Oysaki sprint içinde tamamlanmış ve tüm prensipleri yerine getirmiş bir uygulama yaygınlaştırılabilmelidir. Bu noktada çalışma süresi boyunca tamamlanan işler Product Owner tarafından onaylandıktan sonra zaman kaybetmeden yaygınlaştırılmıştır.

3.17 VERİMLİLİĞİ ARTTIRMAK İÇİN İSRAFI AZALT

Bu pratik yalın yazılım geliştirme çerçevesinden alınmıştır. Yalın yazılım geliştirme yöntemlerinin odaklandığı ortak noktalar; ucuz, hızlı ve çabuk planlama yöntemleri; düşük iletişim masrafları ve etkin şekilde düşük eşgüdüm mekanizmalarıdır.

Bu pratik perspektifinde sprint içinde yapılacak toplantıların zaman kısıtlarının belirlenmesi, geliştirilecek yazılım ürününde tekrarlayan bölümlerin ortak bir standart etrafında toplanıp tekrarların önüne geçerek zamandan tasarruf edilmesi planlanmıştır. Takım, geliştirilecek yazılım projesinde tekrarlamaların önüne geçmek ve bir standart oluşturmak adına standart paketler ile çalışmıştır ve bu çalışma başarılı olmuştur.

3.18 İŞ AKIŞINI GÖRSELLEŞTİRME

İş akışını görselleştirme pratiği Scrum ve Kanban çerçevelerinden alınmıştır. Sprint başında yapılacak işler görevlere ayrılarak not kağıtlarına yazılır ve yapılacak işler panosuna asılır. Geliştirme takım üyesi işi alır ve kâğıda kendi adını yazarak çalışıyorum bölümüne taşır ve iş bittiğinde "BİTTİ" bölümüne taşıyarak işini sonlandırır. Şekil 3.1'de projeye ait çalışma tahtası fotoğrafı görüntülenmektedir.

Şekil 3.1: Sprint İş Listesi Panosu

İşlerin görselleştirilmesi sayesinde, işler eksiksiz tamamlanması ve takım içinde ki şeffaflığın yükseltilmesi amaçlanmıştır. Çalışma boyunca takıma yol haritası olmuştur, çevik metodolojiler için önemli bir yere sahiptir.

3.19 YAPILAN İŞLERİ BELGELE

Çevik metodolojilerin ve kendini tarif eden tüm çerçevelerin üzerinde durduğu nokta az da olsa belgele fikri ile ortak bir zemin bulmaktadır. Belgeleme bölümü birçok kişi tarafından yanlış anlaşılması sebebiyle maalesef geriye dönüşü olmayan bazı yazılım proje tahribatlarına neden olmaktadır. Yalın yazılım geliştirme, Scrum ve XP gibi çerçeveler yeteri kadar belgeleme üzerinde durmaktadır. Yeteri kadar belgeleme bir yazılım ürünün bakımı ve üzerine yapılabilecek yeni özellik eklentilerinde oldukça fazla öneme sahiptir.

Bu bilgiler ışında kendi çalışmamız boyunca her yapılan iş parçacığı için analiz dokümanı, kullanıcı dokümanı, teknik doküman ve yaygınlaştırma dokümanı oluşturma kararı alınmıştır. Proje süresi boyunca tüm doküman oluşturma kurallarına sadık kalınmış ve hem bakımı hem de kullanımı kullanıcı dostu uygulamalar oluşturulmuştur.

3.20 SÜREKLİ GELİŞİM

Sürekli gelişim pratiği MSF çerçevesinden alınmıştır. Takım üyeleri sürekli bilgi paylaşımında bulunarak gelişim içinde olurlar. Örneğin, başlangıçta takıma test mühendisi olarak katılan kişi ilerleyen süreçte kod geliştirmeye, aynı şekilde yazılım geliştiren takım üyesi ise test yapabilir düzeye gelmesi amaçlanmaktadır. Bilgi paylaşımı ile takım içinde sorumluklar eşit düzeyde dağıtılması planlanmaktadır.

Proje için oluşturulan takımda, başlangıçta roller analist, testçi, yazılımcı olarak belirlense de ilerleyen süreçte, yöntemin başarılı olduğu ve sorumlulukların paylaşıldığı görülmektedir. Ayrıca bu durum bilgi paylaşımının da önünü açtığından dolayı takımdan bir kıdemli yazılımcı ayrılmış olsa da bilgi takım içinde paylaşılmış olduğundan yeni gelen personel ile sürece devam edilebilmiştir. Özellikle bilginin bir kişide toplanması

durumu maalesef günümüzde birçok kurumsal şirketin dar boğazlarından biridir. Bazen böyle durumlarda yöneticiye ast atayarak bilgilerini asta yedeklemesi istenir ancak bu durum yönetilebilir ve sürdürülebilir bir yöntem değildir. Çevik yöntemler ile takım çalışmasının üzerinde durulması ve her şirketin kendisine özgü bir çevik çerçeve belirlemesi önem arz etmektedir.

3.21 SPRINT DEĞERLENDİRME TOPLANTISI

Sprint değerlendirme toplantısı Scrum çerçevesinden alınmıştır. Değerlendirme toplantısı, her bir sprintin sonunda ürün parçasını görüp kontrol etmek ve gerekiyorsa ürün iş listesini uyarlamak için düzenlenir. Takım ve paydaşlar bu toplantıda sprintte yapılan işi görüşürler. Bu görüşmeye ve sprint boyunca ürün iş listesinde yapılan değişikliklere dayanarak, katılımcılar değeri en üst seviyeye çıkarmak adına yapılabilecekleri belirlemek için iş birliği yaparlar. Bu gayrı resmî bir toplantıdır, bir durum tespiti toplantısı değildir. Ürün Parçasını sunmanın amacı geribildirim almak ve iş birliğini artırmaktır (Schwaber ve Shutherland, 2013)

Takımın tüm değerlendirme toplantılarına şirket içinde yazılım ürünüyle ilişkili olan departmanlardan yetkililer davet edilmiştir ve kendilerinden alınan geri bildirimler ile daha kaliteli yazılım ürünleri ortaya çıkarılmıştır.

3.22 SPRINT RETROSPEKTİF (GEÇMİŞİ DEĞERLENDİRME) TOPLANTISI

Sprint retrospektif toplantı pratiği Scrum çerçevesinden alınmıştır. Bu toplantılar sprint sonlarında gerçekleştirilir. Son sprintteki ilişkiler, süreç ve araçlar bakımından sürecin gözlemlenmesi amaçlanmaktadır. Mevcutta iyi giden işler ve iyileştirmeye açık konular üzerinde durulur, takım daha iyisini nasıl yaparız konusunu üzerinde tartışır ve süreci iyileştirebilecek bir plan oluşturur.

3.23 KABUL TESTİ

Kabul testi pratiği ASD çerçevesinden alınmıştır. Sprint boyunca alınan işlerin geliştirme takım üyeleri tarafından tamamlanması sonunda Product Owner tarafından yapılmaktadır. Product Owner takımdan talep ettiği işin, karşılanıp karşılanmadığını kabul testi yaparak gözlemler. Eğer istenildiği gibi bir geliştirme yapılmış ve hata ile karşılaşılmamış ise yaygınlaştırılması yapılır.

4. ÇEVİK YAZILIM GELİŞTİRME ÇERÇEVESİ: PROJE UYGULAMASI

Perakende sektör dinamiklerine ve projeye uygun bir çevik yazılım geliştirme çerçevesi belirlendikten sonra çerçevenin başarısını ölçümlemek adına bir projede uygulanması planlanmıştır. Bu bilgiler ışığında, şirketin tedarik zinciri iş biriminden proje gereksinimleri üzerine bir toplantı gerçekleştirilmiştir. Bu toplantıda, şirketin dağıtım merkezlerinde kullanılmak üzere akıllı bir adres bazlı stok yönetim sisteminin geliştirilmesi kararlaştırılmıştır. Bu yeni sistemin mobil cihazlar üzerinde çalışılabilir olması, mevcutta kullanılan ERP sistemine entegre edilmesi gibi teknik birtakım ihtiyaçlar talep edilmiştir.

Uygulama geliştirme aşamasına geçmeden önce yeni çevik çerçeve pratikleri üzerinden yazılım geliştirme takımı oluşturulmuştur. Takım, üç yazılım geliştirme personeli (biri analist yazılımcı), bir testçi ve iş biriminden bir iş sahibi (Product Owner) ile toplamda beş kişiden oluşmaktadır. Takımın dışarıdan gelebilecek müdahalelere ve konsantrasyon bozukluklarını engellemek adına şirketin toplantı odalarından biri tahsis edilerek, uygun araç ve gereçlerle donatılmıştır.

Takım oluşumu ve fiziksel şartlar yerine getirildikten sonra, takıma yeni çevik çerçeve hakkında gerekli eğitimler iki gün boyunca verilmiştir. Eğitimler sonrasında, sprint-0 adı altında bir hazırlanma iterasyonu oluşturularak, üyelerin yeni pratiklere uyum göstermesi ve alışması hedeflenmiştir. Sprint-0 sürecinde, yazılım projesinin hayata geçirilmesi için gereken işler gözden geçirilmiş ve her bir işe puanlandırma yapılmıştır. Bu puanlandırma ve süre tahminleme sonucunda projenin toplamda 4 ay, yani 8 sprint sürebileceği belirlenmiştir. 8 sprint olarak belirlenmesinin nedeni takımın her bir sprinti 2 haftalık yinelemeler ile ilerlemeye uygun görmesinden kaynaklanmıştır. Nitekim, 1 haftalık yineleme oldukça kısa, 3 haftalık yineleme ise klasik yazılım geliştirme metodolojilerini çağrıştırdığından dolayı uzun olarak görülmüştür.

Projenin hayata geçirilmesi sürecinde uygulanan aşamalar:

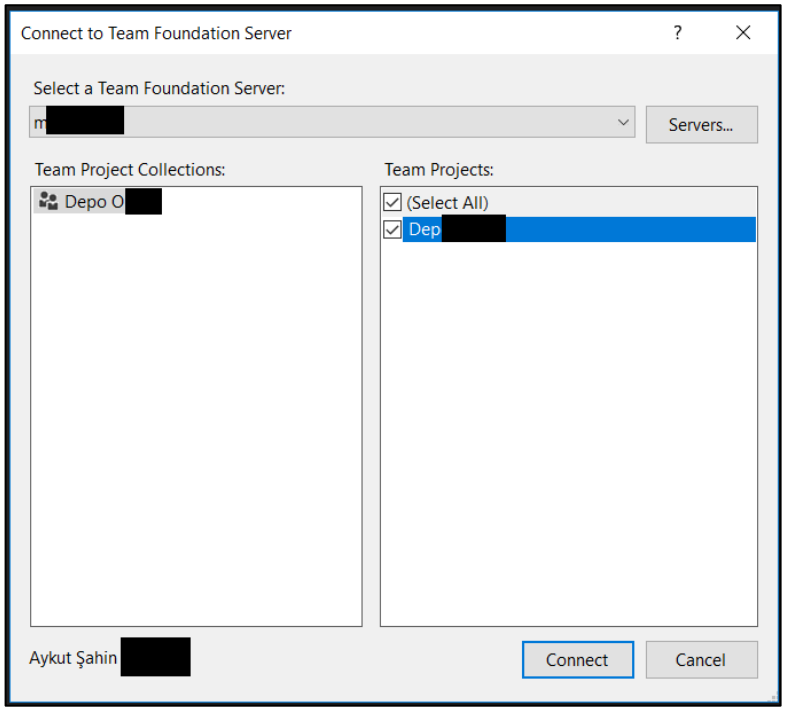
4.1 TAKIMIN ÇALIŞMA ORTAMININ BELİRLENMESİ VE PROJE İÇİN KURULUMLARIN YAPILMASI

Projede uygulama geliştirme platformu olarak, şirketin kurumsal çözümlerinde kullanılan Oracle Forms uygulama geliştirme platformu ve Oracle Veri tabanı kullanılması planlanmıştır. Uygulama versiyonlarının takip edilmesi amacıyla Microsoft Visual Studio Team Foundation Server kullanacaktır. Ayrıca iş sahibi tarafından takıma getirilecek işlerin ve takımın aldığı işleri görevlere ayırırken Microsoft Excel kullanması kararlaştırılmıştır. Yeni çevik yazılım geliştirme çerçevesinin başarılı olması durumunda, bu çerçeveye uygun bir uygulama çözümü geliştirilmesi planlanmıştır.

4.2 TEAM FOUNDATİON SERVER (TFS) KURULUMU VE PROJE GRUBU TANIMLAMALARININ YAPILMASI

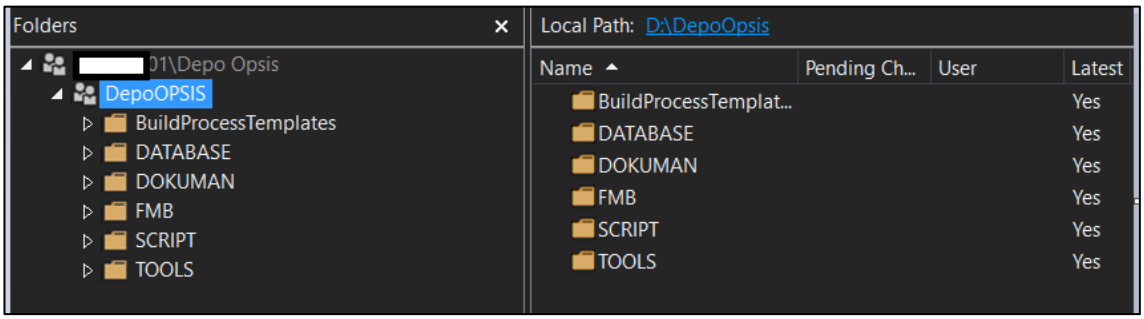
Mirttso4501/DepoOp adında yeni bir server ve dizin belirlenmiştir. Proje boyunca tüm yapılacak geliştirmeler bu dizin altında versiyonları yapılarak depolanmıştır. Şekil 4.1'de Team Foundation Server üzerinde çalışma dizini oluşturulması gösterilmiştir.

Şekil 4.1: TFS Üzerinde Çalışma Dizini Tanımlama



Çalışma dizini altında bulunması gereken klasörler Şekil 4.2'de gösterilmektedir. Proje boyunca geliştirilecek tüm uygulamalar, dizin altında "FMB" bölümünde versiyonlaması yapılıp, depolanmıştır. Veri tabanı üzerinde geliştirmeler "DATABASE" klasöründe, uygulamalara ait dokümanlar ise "DOKUMANLAR" dizininde saklanmıştır.

Şekil 4.2: Versiyon Uygulaması Dizin Atamaları



4.3 PRODUCT BACKLOG LİSTESİNİN PRODUCT OWNER TARAFINDAN OLUŞTURULMASI

Dağıtım Merkezlerinde uygulanmak üzere geliştirilecek adres bazlı stok yönetim sisteminin gerekliliklerini ve ihtiyaçlarının İş Listesi (Product Backlog) listesine başlıklar halinde Product Owner tarafından yazılması istenmiştir. Product Backlog ve ilerleyen bölümde anlatılacak Görev Listesi (Sprint Backlog) dosyaları oluşturulması için Microsoft Excel kullanılması kararlaştırılmıştır.

Şekil 4.3: Orijinal Product Backlog Listesi Görüntüsü

PRODUCT BACKLOG

PPM No	No1	No2	Item	B.V.	Size	B.V./ Size	Sprint Commited	Priority	Status
	110	0	Adres bazlı stok projesi (faz1)						
82280	110	1	Adres bazlı stok - için forklift operatoru mal yerleştirme, indirme, yer değiştirme uygulamasının yapılması	220	55	4,00	1	84	DONE
82280	42	6	Adres bazlı stok - forkliftlerde kullanılmak üzere tablet uygulamasının oluşturulması	100	21	4,76	1	96	DONE
82716	110	5	Adres Bazlı Stok - kapsamında PC üzerinden anlık raporlama ile tüm ilişkilendirilmiş palet barkodları listelenebilmelidir. Adres, palet barkodu, mal barkodu, mal numarası, miktar bilgisi, yerleştiren kullanıcı, palet oluşturma tarihi ve son kullanma tarihi listelenmelidir.	180	21	8,57	2	96	DONE
83257	110	4	Adres bazlı Stok envanter sorumlularının kullanacağı yanlış adres- plt ilişkilerinin düzeltilebileceği aynı zamanda bilgilendirme sağlayacak bir ekran gerekli olacaktır. Adres okutulduğunda paleti, palet okutulduğunda adresi ve içerikteki bilgileri gösterir. Yeni adres kısmı ile de gerçek adres okutularak data düzeltilebilir, ekstra olarak adres bosalt adımı eklenebilir.	180	34	5,29	2	99	DONE

Product Backlog listesi tüm kullanıcılar tarafından görüntülenebilir ancak sadece Product Owner tarafından değiştirilebilmektedir.

4.4 SPRİNT BACKLOG LİSTESİNİN GELİŞTİRME TAKIM ÜYELERİ TARAFINDAN OLUŞTURULMASI

Planlama toplantısı sonrası alınan işler, yeni oluşturulan çevik çerçeve ilkelerine dayanarak görev detaylarına ayrılır. Bu detaylar, Sprint Backlog dosyasına yazılır, Şekil 4.4'de bu projeye ait orijinal Sprint Backlog dosyası görülmektedir, tüm bu dosyaya ait düzenleme ve eklemelerden geliştirme takım üyeleri sorumludur.

Şekil 4.4: Orijinal Sprint Backlog Dosya Görüntüsü

PBI #	PBI Size	PBI Prty	Task ID	57	Task Detail	PBI DONE Day	Performer(s)	
46	21,00	1,0	1		Adres bazlı stok için forklift operatoru mal yerleştirme, indirr	55	DONE	
			2		Rafa mal yerleştirmetablet uygulamasının create edilmesi			20
			3		Palet yerleştirme özelliğinin oluşturulması			15
			4		Palet indirme özelliğinin oluşturulması			22
			5		Palet yer değiştirme özelliğinin oluşturulması			20
			6		Analiz dokümanının oluşturulması			4
			7		Code review			7
			8		Unit Test			3
			9		Fonksiyonel test			5
			10		UAT destek			3
			11		Tüm Doküman Oluşturulması			8
			12		Yaygınlaştırma Notu			1
2.1	13,00	1,0	13		Adres Bazlı Stok - kapsamında PC üzerinden anlık raporlama	21	DONE	
			14		Adres Bazlı Stok - kapsamında PC üzerinden anlık raporlama oluşturulması			2
			19		Analiz dokümanının oluşturulması			3
			20		Code review			5
			21		Unit Test			4

4.5 ADRES BAZLI STOK YÖNETİM SİSTEMİNİN (ABS), YENİ ÇEVİK ÇERÇEVE İLE GELİŞTİRİLMESİ

Adres bazlı stok yönetim projesinde ki temel amaç, dağıtım merkezlerine kabulü yapılan ürünlerin paletlenmesi, ilgili raflara yönlendirilmesi ve raflarda bulunan paletlerin ürün bakiyelerinin takip edilmesi üzerine kurgulanmıştır. Bu sayede dağıtım merkezi içinde herhangi bir ürüne ait stok ve stokların yerleşim planı izlenebilir hale gelecektir.

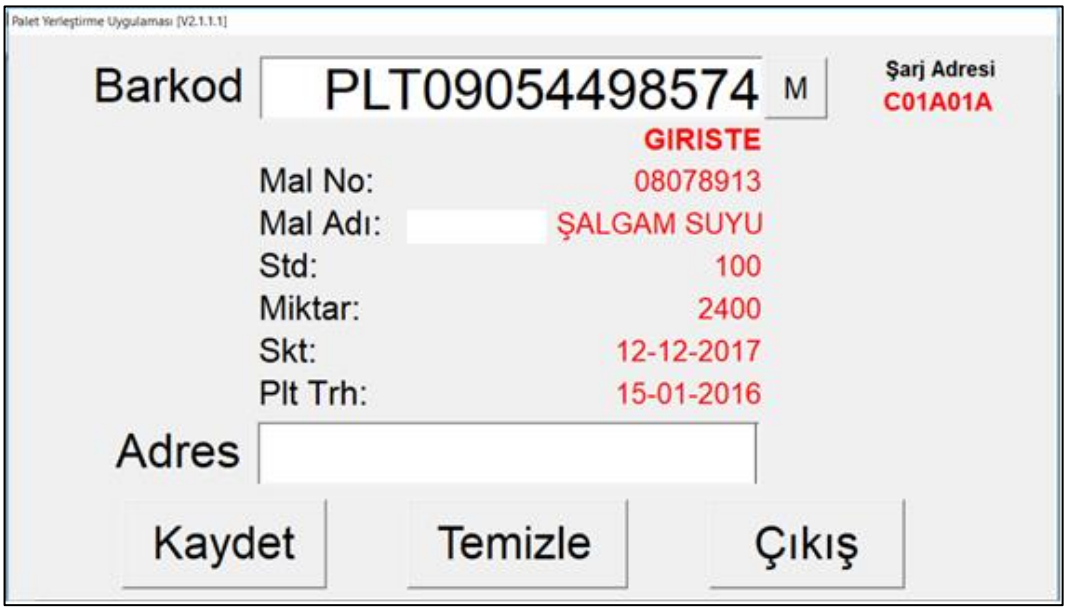
Yazılım çözümü, ürünü raflara taşıyan forklift operatörünün kullanacağı tablet uygulaması, rafları denetleyen denetçinin kullanacağı el terminali uygulamaları, raf etiket basım uygulaması ve raporlamalar ile tamamlanması planlanmıştır. Sırasıyla proje için geliştirmesi yapılan uygulamalar:

4.5.1 ABS Projesi Tablet Uygulamalarının Oluşturulması

Dağıtım merkezine kabulü yapılan ürünlerin paletlenme aşamasından sonra ilgili adrese yönlendirilmesi için kullanılan uygulamadır.

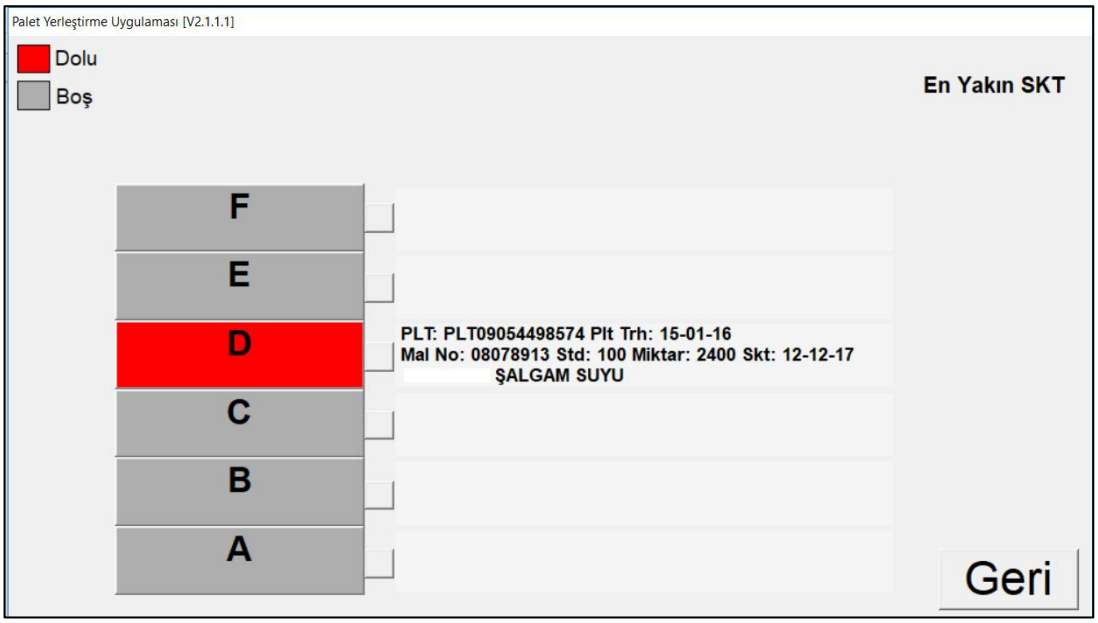
Uygulama tablette, forklift operatörü tarafından kullanılacağından dolayı oldukça anlaşılabilir ve işlevsel olması üzerinden durulmuştur Şekil 4.5'te görüleceği üzere forklift operatörü taşıyacağı paletin palet etiketini, etiket okuyucu yardımı ile okutur, uygulama ilgili paletin içinde bulunan ürün ve miktar bilgilerini ekranda görüntüler ve operatörü ürüne ait ilgili şarj adresine yönlendirir.

Şekil 4.5: Palet Yerleştirme Uygulaması Arayüzü



Forklift ilgili şarj adresine geldiğinde, operatör raf üzerinde bulunan şarj adresini okutur ve uygulama aşağıda Şekil 4.6'da görüldüğü üzere mevcut kat planını ekranda görüntüler. Forklift operatörü uygun olan bir kata paleti yerleştirir ve operasyon tamamlanır.

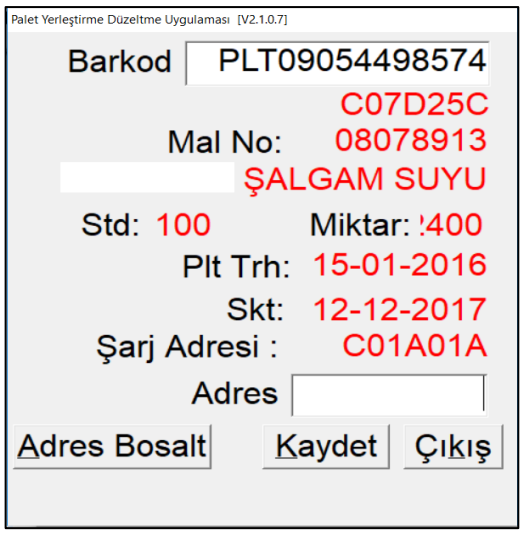
Şekil 4.6: Palet Yerleştirme Uygulaması Kat Planı Arayüzü



4.5.2 ABS Projesi El Terminali Uygulamalarının Oluşturulması

Bölüm 3.5.1'de anlatılan forklift operatörünün kullandığı tablet uygulamasına alternatif olması amacıyla el terminallerinde kullanılmak üzere Şekil 4.7'de ara yüzü gösterilen palet yerleştirme uygulaması yapılmıştır.

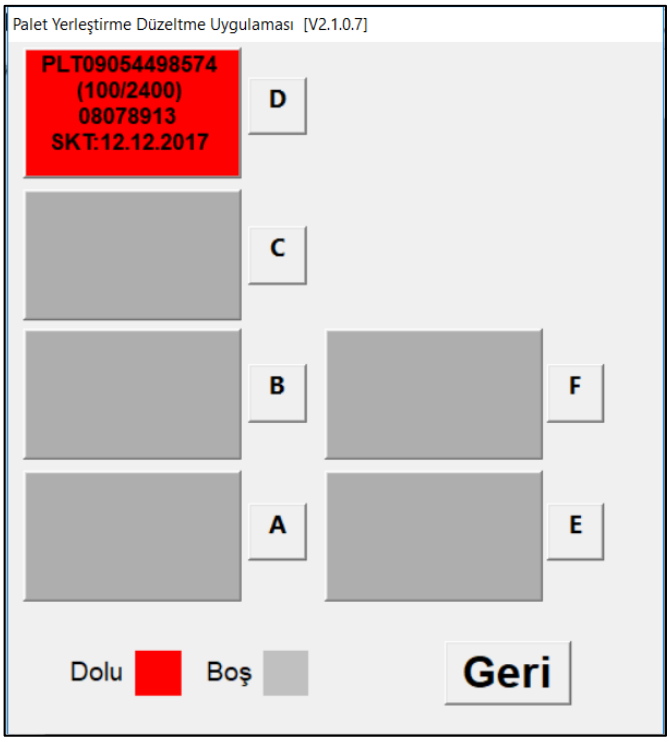
Şekil 4.7: Palet Yerleştirme El Terminali Uygulaması



Tablet uygulaması ile tüm işlevsellikleri aynı olmakla birlikte, yetkili personel tarafından kullanılması planlanarak adres boşaltma özelliği eklenmiştir. Kullanıcı hatalı olduğunu gördüğü adresi, adres etiketini okutup, ilgili katı seçerek, adres boşalt butonunu kullanarak düzeltebilecektir.

Şekil 4.8'de el terminali uygulaması kat planı görüntülenmektedir, el terminali çözünürlüğü gereği ortalama 6 kat olan rafların, dördü solda ikisi sağda olmak üzere yan yana planlaması yapılmıştır.

Şekil 4.8: Palet Yerleştirme El Terminali Kat Planı



4.5.3 ABS Projesi Palet Düzenleme Uygulaması

Proje kapsamında, palet içi olası miktarsal hataları gidermek ve yetkili personellerin kullanması amacıyla el terminalleri üzerinde palet düzeltme uygulaması yapılmıştır.

Şekil 4.9: Palet Düzeltme Uygulaması Arayüzü

Şekil 4.9'da görüleceği üzere düzeltme yapılacak palet etiketi okutulduktan sonra düzelt butonuna tıklanır. Şekil 4.10'daki ekran vasıtasıyla ilgili palete ait miktarsal düzeltmeler artı veya eksi yönde yapılabilir.

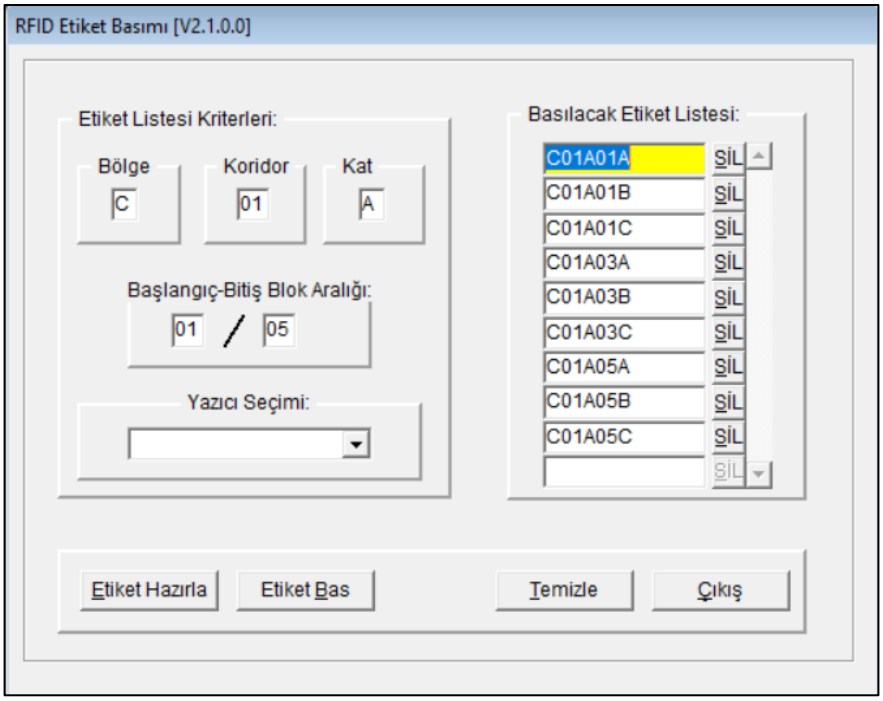
Şekil 4.10: Palet Düzeltme Miktar Uygulaması

4.5.4 ABS Projesi Kapsamında Adreslerin Etiketlenmesi

Adres bazlı stok yönetimi kapsamında etiketlerin adreslemesi amacıyla, etiket basım uygulaması oluşturulmuştur.

Şekil 4.11'de görüleceği üzere, etiketler dağıtım merkezinin adresleme kriterlerine uygun, bölge, koridor, kat ve başlangıç bitiş blok aralıkları kullanılarak yapılmıştır. Etiketler Zebra etiket printer kullanılarak yazdırılmış ve ilgili raflara yapıştırılmıştır.

Şekil 4.11: Adres Etiketleme Uygulaması Arayüzü

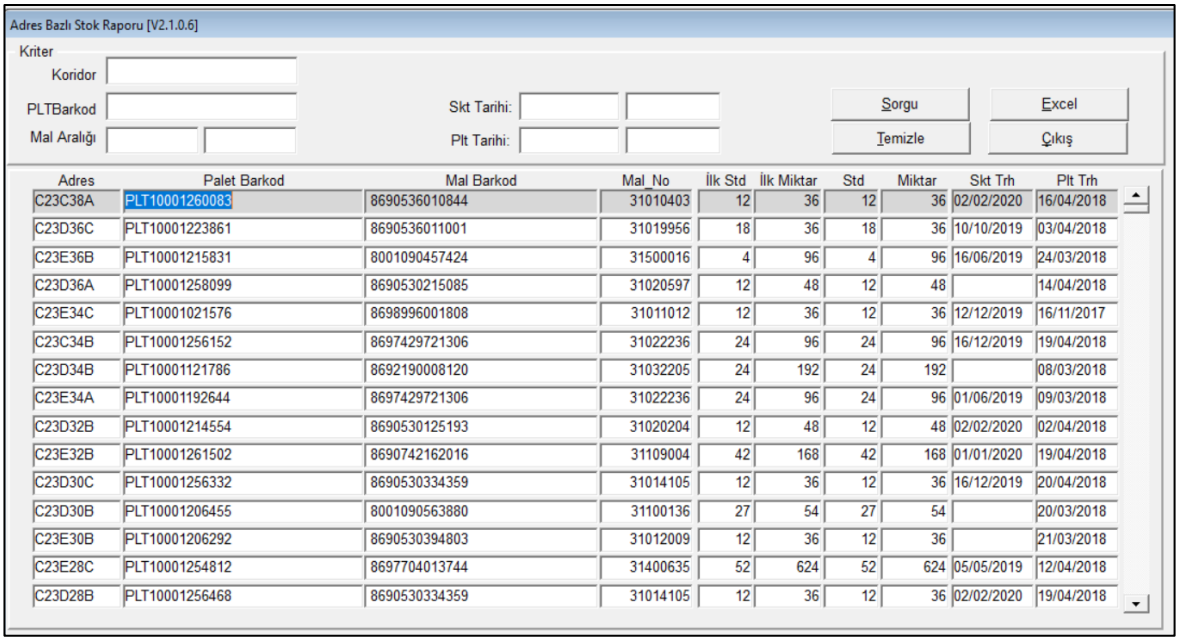


4.5.5 ABS Projesi Raporlama Uygulamalarının Oluşturulması

Projede haritalanan stok ve şarj adreslerinde hangi palet ve ürünlerin olduğu, miktarsal bakiye durumları ve son kullanma tarihi vb. özelliklerinin raporlandığı Şekil 4.12' de görüntülenen rapor uygulaması oluşturulmuştur.

Bu uygulama sayesinde tüm adresler, ürün bakiyeleri ve toplam miktarları vb. bilgiler ulaşılabilir olmuştur.

Şekil 4.12: ABS Rapor Uygulaması Arayüzü



Adres	Palet Barkod	Mal Barkod	Mal_No	İlk Std	İlk Miktar	Std	Miktar	Skt Trh	Plt Trh
C23C38A	PLT10001260083	8690536010844	31010403	12	36	12	36	02/02/2020	16/04/2018
C23D36C	PLT10001223861	8690536011001	31019956	18	36	18	36	10/10/2019	03/04/2018
C23E36B	PLT10001215831	8001090457424	31500016	4	96	4	96	16/06/2019	24/03/2018
C23D36A	PLT10001258099	8690530215085	31020597	12	48	12	48		14/04/2018
C23E34C	PLT10001021576	8698996001808	31011012	12	36	12	36	12/12/2019	16/11/2017
C23C34B	PLT10001256152	8697429721306	31022236	24	96	24	96	16/12/2019	19/04/2018
C23D34B	PLT10001121786	8692190008120	31032205	24	192	24	192		08/03/2018
C23E34A	PLT10001192644	8697429721306	31022236	24	96	24	96	01/06/2019	09/03/2018
C23D32B	PLT10001214554	8690530125193	31020204	12	48	12	48	02/02/2020	02/04/2018
C23E32B	PLT10001261502	8690742162016	31109004	42	168	42	168	01/01/2020	19/04/2018
C23D30C	PLT10001256332	8690530334359	31014105	12	36	12	36	16/12/2019	20/04/2018
C23D30B	PLT10001206455	8001090563880	31100136	27	54	27	54		20/03/2018
C23E30B	PLT10001206292	8690530394803	31012009	12	36	12	36		21/03/2018
C23E28C	PLT10001254812	8697704013744	31400635	52	624	52	624	05/05/2019	12/04/2018
C23D28B	PLT10001256468	8690530334359	31014105	12	36	12	36	02/02/2020	19/04/2018

Projeye ait tablo tasarımları ve ana verilerin oluşturulması, bu tezin sonunda yer alan ekler bölümünde Ek A1, Ek A2 ve Ek A3 bölümünde anlatılmıştır.

Projenin temel fonksiyonları ve algoritması PL/SQL ile Oracle veri tabanında SP_ABS_ISLEMLERI adı altında bir pakette toplanmıştır. Pakette bulunan fonksiyonların açıklamaları Ek A4 bölümünde yer almaktadır.

5. SONUÇ VE DEĞERLENDİRME

Yeni çevik çerçevenin, tedarik zinciri yönetimi iş birimine ait adres bazlı stok yönetimi projesine uygulanması için üç yazılımcı (biri analist yazılımcı), bir testçi ve bir iş sahibi ile başlanmıştır. Projenin 3. Sprinti içinde bir yazılımcı işten ayrılmış yerine yeni bir yazılımcı dahil edilmiştir. Bu süreçte takıma yeni katılan yazılımcıya, çevik çerçeve ve pratikleri hakkında detaylı bir eğitim analist tarafından verilmiştir. Takıma yeni dahil olan personel geçmişinde daha bireysel çalıştığından dolayı başlangıçta özellikle takım çalışmasına adapte olmakta güçlük çekse de ilerleyen süreçte adapte olmayı başarmıştır. Ayrıca takım çalışmasının yararlarını gördükçe bu konu hakkında geri bildirimler yapıp takım birlikteliğini artıran davranış biçimi göstermiştir.

Proje başlangıcında Product Owner tarafından hazırlanan ürün listesi, tümüyle geliştirme takımı tarafından ele alınmış ve proje için toplamda 8 Sprintlik yani 4 aylık bir süreç öngörülmüştür. Ancak projenin ilerleyen sprintlerinde uygulamalara gelen ilave özellikler ve operasyonel destek çalışmaları ile birlikte 2 ek sprint yapılarak toplamda 5 aylık bir çalışma süresinde işler tamamlanmıştır. 5 aylık geliştirme süresince 4 geliştirme takım üyesi ve bir iş birimi (iş birimi yüzde 20 zamanını harcamıştır) ile toplamda 2445 saatlik iş gücüyle proje tamamlanmıştır.

Sprint bazında incelendiğinde, toplamda 10 Sprint ve 51 Product Backlog Item, 612 Sprint Backlog Item ve 40 adet operasyon talep oluşturulmuş. Projeye ait sprint detayları Tablo 5.1’de görüntülenmektedir.

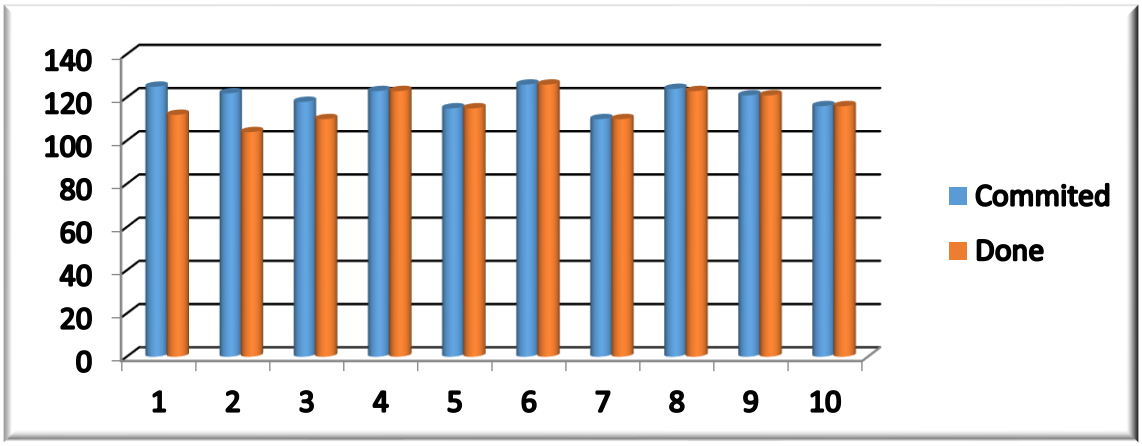
Tablo 5.1: Proje Sprint Detayları

Tarih	Sprint	Periyot	Sprint Büyüklük	Toplam Saat	Product Owner Saat
02.10.17- 13.10.17	**1**	**2 hafta**	**125**	**245**	**18**
16.10.17- 27.10.17	**2**	**2 hafta**	**122**	**242**	**13**
30.10.17- 10.11.17	**3**	**2 hafta**	**118**	**230**	**15**
13.11.17- 24.11.17	**4**	**2 hafta**	**123**	**240**	**16**
27.11.17- 08.12.17	**5**	**2 hafta**	**115**	**220**	**15**
11.12.17- 22.12.17	**6**	**2 hafta**	**126**	**250**	**13**
25.12.17- 05.01.18	**7**	**2 hafta**	**110**	**198**	**8**
08.01.18- 19.01.18	**8**	**2 hafta**	**124**	**243**	**10**
22.01.18- 02.02.18	**9**	**2 hafta**	**121**	**237**	**14**
05.02.18- 16.02.18	**10**	**2 hafta**	**116**	**202**	**16**
TOPLAM:			**1199**	**2307**	**138**

Detayları incelendiğinde sprintlerde yapılacak işlerin geliştirme takım üyeleri tarafından yeni çevik çerçeve ilkelerinden iş büyüklüğü belirleme ilkesi ile sprint büyüklükleri ölçülmeye çalışılmıştır. Proje süresi boyunca koşulan 10 sprint incelendiğinde toplamda 1199 sprint büyüklüğü, ortalama olarak ise 120 sprint büyüklüğüne erişilmiştir. Ortalama sprint büyüklüğü verisi takımın ilerde yapabileceği yeni projelere de ışık tutarak sürdürülebilir kapasitesini ortaya koymaktadır.

Takım, projenin ilk fazı olan ilk 3 sprintte, sprint büyüklerini hesaplama da hata yapmıştır. Bu durum çevik çerçevenin ilkelerinin benimsenmesi ve yapılan işin daha iyi anlaşılması ile beraber ilerleyen sprintlerde daha doğru tahminler yaparak hata oranları düşürmüş, bazı sprintlerde ise hata oranı sıfıra indirilmiştir.

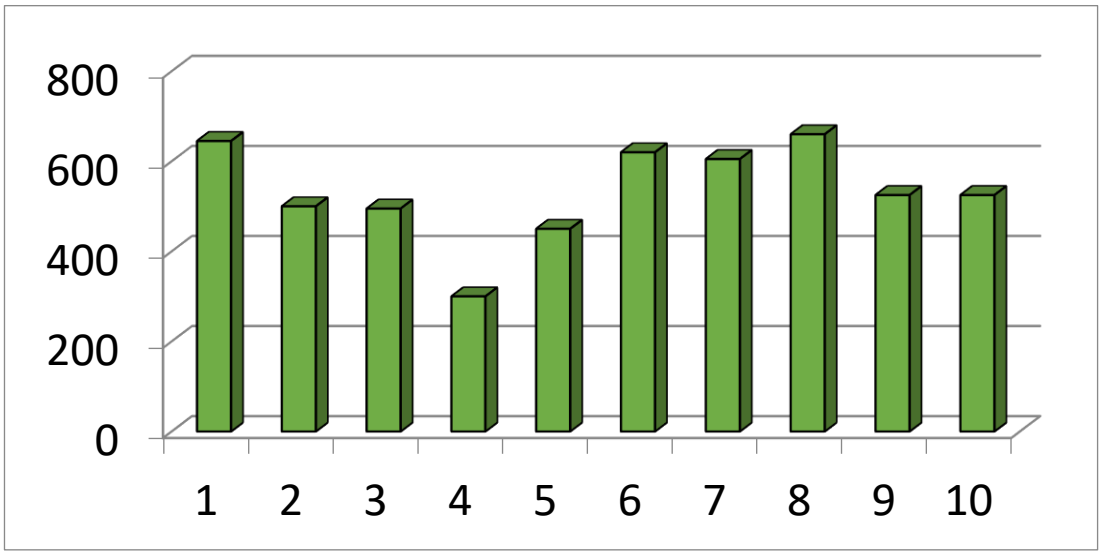
Şekil 5.1: Sprint Tamamlanma Oranları



Sprintlerin tamamlanma oranları incelersek ilk 3 sprintte yüzde 80'lerde kalan tamamlanma oranları 4. Sprint sonrasında yüzde 100 çıkmış, sadece 8. Sprintte yüzde 98'e düşmüştür. Bu veriler ışığında takımın yeni çevik çerçeve ile iş yapma bilgi ve becerisinin 4. Sprint sonunda tamamen oturduğu ve başarılı olduğunu söyleyebiliriz.

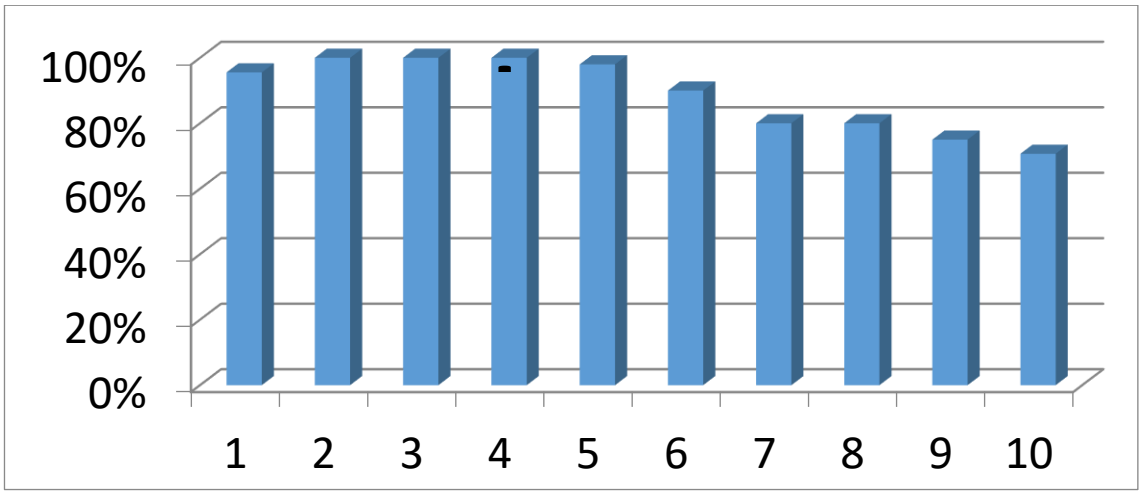
Geliştirme takım üyelerinin gözünde ki iş büyüklükleri ile Product Owner gözünden iş büyükleri farklılık göstermektedir. Bu nedenle iş birimi tarafından yeni çevik çerçeve kapsamında işlerin Sprint bazında büyüklükleri için Product Owner'a danışılmış ve her bir senaryoya değer verilmesi istenmiştir. Şekil 5.2'de iş birimi gözünden sprintlere verilen değerler gösterilmektedir.

Şekil 5.2: İş Değeri (Business Value)



Sonuçlar toplam iş değeri bazında incelediğinde, takımın yüzde 80 ve yüzde 98 tamamlanma oranında 1. ve 8. Sprintlerin toplam değerinin diğerlerinden daha fazla olduğu gözlemlenmiştir. Buradan sonuçla, bir işin sistem analizi ve yazılımının karışık ve karmaşık olması, bu işi yazılım geliştirme penceresinden büyük olarak gösterse de aynı işin iş birimi tarafından gözüken sağladığı katma değerinin, yazılımsal karmaşıklığına oranla çok daha az olabileceği gözlemlenmiştir. Bu durum projenin ilk sprintlerinde Product Owner ile Geliştirme Takım Üyeleri tarafından tartışmalara neden olmuştur. Ancak yeni çevik yöntemin yaklaşımları olan yüz yüze iletişim, şeffaflık ve çalışan yazılım gibi iyi niyet ortamını güçlendiren ilkeleri nedeniyle kısa sürede çözüme ulaşmıştır.

Şekil 5.3: Inovasyon (Yenilik) Oranı



Sprintler boyunca yapılan işlerde ki yenilikçi yaklaşım bir başka değişle inovasyon durumunun kontrolü yapılmıştır. Projenin ilk sprintlerinde ki inovasyon oranlarının özellikle son 3 sprintten çok daha yüksek olduğu gözlemlenmiştir. Bunun nedeni ise projenin ilk safhalarında geliştirilen hemen hemen tüm özelliklerin inovasyon oluşturulmasıdır, son sprintlerde ise mevcut oluşturulan uygulamaların üzerine yeni ek özellikler ve mevcut yazılımın hata oluşturan parçalarının onarılması için zaman harcanmıştır. Yeni çevik çerçeve kapsamında her bir sprint için ilk 3 sprintte operasyonel olarak, yani içinde uygulama düzeyinde destek, bug-fix vb. konular için ne kadar zaman harcandığı ölçülmüştür. Bu oran aslında takımın ne düzeyde kaliteli yazılım ürünü ortaya çıkardığının da bir göstergesidir. Proje süresi boyunca bug-fix oranı 8 büyüklüğü

geçmemiştir. Bu oran, toplam ortalama sprint büyüklüğünün 120 olduğu düşünülürse yüzde 6,6 olarak ölçülmüştür. Yeni çevik çerçeve oluşturulurken yapılan araştırmalarda yüzde 10'luk bug-fix oranının normal kabul edildiği düşünüldüğünde yüzde 6,6 ortalamanın oldukça altında kaldığı gözlemlenmiştir ve bu durum yeni çevik çerçevenin ve takımın başarısı olarak yorumlanabilir.

Mevcut projenin gereklilikleri ve yazılım geliştirilecek pazarın ihtiyaçlarına göre çevik metodolojik yaklaşım ve bugüne kadar oluşturulmuş çevik çerçevelerin uygun ilkeleri bir araya getirilerek çok daha güçlü bir çerçeve oluşturmak, proje başarısını artıracak ve ihtiyacı en doğru biçimde karşılayacak yazılım ürünleri ortaya çıkaracaktır.

Perakende sektörünün hızlı değişimine ve tedarik zinciri yönetiminin karmaşık zincir yapısına uygun proje başlangıcında oluşturulan yeni çevik çerçeve sayesinde klasik proje yönetim yaklaşımları ile 9-10 ay olarak belirlenen projenin 5 ay gibi yarı sürede tamamlanması sağlanmıştır. Maddeler halinde yeni çevik çerçevenin ve iş yapış şeklinin avantajlarını incelersek;

i. İş biriminin gözünde değerli olan yazılımın erken ve sürekli teslimatı ile müşteri memnuniyeti artması,
ii. Takım üyelerinin bir arada, bir masa etrafında toplanması ile ortaya çıkan sorunların en kısa sürede çözüme kavuşturulması,
iii. Proje başlangıcında ekip olarak ortaya çıkan ve bireysel çalışmaya alışmış ekip üyelerinin zamanla takıma dönüşmesi ve birbirlerine kenetlenmeleri,
iv. Takımın sürdürülebilir kapasitesinin belirlenmesi sayesinde projenin ilerleyen sürecinde daha iyi planlama yapma yeteneğinin ortaya çıkması
v. Yapılan işlerin sürekli izlenebilir olması nedeniyle projenin aşamalarının tüm safhalarıyla net bir biçimde görüntülenebilmesi,
vi. Yapılan tüm işlerin takım içinde dağıtılması sayesinde hem takım içi hem de dışarıdan şeffaflığın en üst derecede yaşanması,
vii. İş biriminden bir kişinin takıma monte edilmesi ile işlerin her an denetlenmesi ve ola ki bir yanlış anlamanın anında tespit edilip düzeltilmesinin sağlanması,

viii. Sprint içinde yapılacak işlerin hangi amaca hizmet ettiği, yararları ve kazanımlarının net bir biçimde takım içinde paylaşılması sayesinde motivasyonun yüksek olması,

ix. Tam zamanında teslim yapabilme kabiliyetinin oluşması,

x. İş birimi tarafından getirilen değişikliklerin projenin son aşamasında dahi olunsa kabul edilebilir olması,

xi. Takım üyelerinin yüz yüze iletişimi sayesinde, doğru bilginin paylaşımının ve ortaya çıkabilecek sorunların hızlı biçimde çözümünün sağlanması

xii. Kalite beklentisi ve müşteri memnuniyeti sağlanması amacıyla en iyi araçlar ve yetkinlikleri yüksek bireylerle çalışılması

KAYNAKÇA

Kitaplar

Highsmith, J., 2000. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems.* 1. Baskı. New York: Dorset House Publishing Co.

Larman, C., 2007. *Agile and Iterative Development: A Manager's Guide*. 1. Baskı. Boston: Addison-Wesley, Pearson Education, Inc.

McConnell, S., 2015. *Code Complete: A Practical Handbook of Software Construction, Second Edition.* 24. Baskı. Washington: Microsoft Press.

Palmer, S. R. & Felsing, M. J., 2001. *A Practical Guide to Feature-Driven Development.* 1. Baskı. New Jersey: Prentice Hall.

Poppendieck, M. & Poppendieck, T., 2013. *Lean Software Development: An Agile Toolkit*. 17. Baskı. Indiana: Addison-Wesley, Pearson Education, Inc.

Reddy, A, 2015. *The Scrumban [R]Evolution.* 1. Baskı. Indiana: Addison-Wesley, Pearson Education, Inc.

Schwaber K., 2004. *Agile Project Management With Scrum.* 1. Baskı. Washington: Microsoft Press.

Turner M. S. V., 2006. *Microsoft Solutions Framework Essentials*. 1. Baskı. Washington: Microsoft Press.

Womack, J.P., Jones, D.T., Roos, D, 2007. *The Machine That Changed the World: The Story of Lean Production.* 1. Baskı. New York: Free Press.

Womack J.P., Jones, D.T., 2003. *Lean Thinking: Banish Waste and Create Wealth in your Corporation.* 2. Baskı. New York: Free Press.

Süreli Yayınlar

Abrahamsson, P., Salo, O., Ronkainen, J. ve Warsta, J., 2002. Agile software development methods review and analysis. *VTT Publications*, **478**, ss. 3–107.

Beck, K., 1999. Embracing change with extreme programming. *IEEE Computer*, **32**(10), ss. 70–77.

Bissi, W., Serra Seca Neto, A. G. ve Emer, M. C. F. P., 2016. The effects of test driven development on internal quality, external quality and productivity: A systematic review. *Information and Software Technology*, **74**, ss. 45–54.

Campanelli, A. S. & Parreiras, F. S., 2015. Agile methods tailoring – A systematic literature review. *Journal of Systems and Software*, **110**, ss. 85-100.

Cohen, D., Lindvall, M. ve Costa, P., 2004. An Introduction to Agile Methods. *Advances in Comp*., **62**, ss. 1–66.

Flora, H. K. ve Chande, S. V., 2014. A Systematic Study on Agile Software Development Methodologies and Practices. *International Journal of Computer Science and Information Technologies (IJCSIT)*, **5**(3), ss. 3626–3637

Millington, D. ve Stapleton, J., 1995. Developing a RAD standard. *IEEE Software*, **12**(5), ss. 54–55.

Diğer Yayınlar

Abrahamsson, P., Warsta, J., Siponen, M. T., Ronkainen, J. ve Ronkanen, J., 2003. New Directions on Agile Methods: A Comparative Analysis. *Proceeding of the 25th International Conference on Software Engineering*, ss. 244–254

Ahmad, M. O., Markkula, J. ve Ovio, M., 2013. Kanban in Software Development: A Systematic Literature Review. *Software Engineering and Advanced Applications (SEAA), 39th EUROMICRO Conference*, ss. 9–16.

Agilemodellling.com, *Disciplined Agile Software Development: Definition,* http://agilemodeling.com/essays/agileSoftwareDevelopment.htm. [erişim tarihi 20 Ocak 2018].

Beck, K., Cockburn, A., Jeffries, R. ve Highsmith, J., 2001. Çevik Yazılım Geliştirme Manifestosu, http://agilemanifesto.org/iso/tr/manifesto.html. [erişim tarihi 03 Şubat 2017]

History: The Agile Manifesto, http://agilemanifesto.org/history.html. [erişim tarihi 03 Şubat 2017].

Keeton, M., Turner, M., 2006. Wikipedia. [Çevrimiçi] https://en.wikipedia.org/wiki/Microsoft_Solutions_Framework [erişim tarihi 12 Aralık 2017].

Schwaber K., Sutherland J., 2013. *Scrum Tanımlayıcı Klavuzu: Oyunun Kuralları.* http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-tr.pdf. [erişim tarihi 11 Ekim 2017].

Watts, J., 2017. *Agile Method of Software Development, Blog.* https://number8.com/kanban-versus-scrum/ [erişim tarihi 10 Nisan 2018].

EKLER

Ek A.1 Projede Kullanılan Tabloların İçerik ve Amaçları

Tablo 6.1: Ürün Adres Stok İlişki Tablosu

URUN_ADRES_STOK
YER_KODU
MAL_ADRES
URUN_NO
ST_MIKTAR
MIKTAR
SON_HAR_TRH
PALET_BARKOD
RFID_RAF_ETIKET
SKT

Amacı: Ürün, palet ve stok adres bilgilerinin ilişkilerinin tutulduğu tablodur. Tablo ayrıca ilk hali itibariyle sahadaki tüm adreslerin bilgilerini içerir. İlerde oluşacak tüm hareketlerde tabloda sadece güncelleme işlemi yapılır. Tablo üzerinden herhangi bir silme işlemi yapılmaz.

Tablo 6.2: Palet Giriş Tablosu

PALET_GIRIS
PBARCODE
MAL_NO
SIM
ST
MIKTAR
ISLEM_TRH
KULLANICI_NO
FIS_TUR
YER_KODU
FIS_NO
FIS_TRH
SKT
ADRES
PARTI_NO
BAKIYE_ST
BAKIYE_MIKTAR
BITTI

Amacı: Dağıtım Merkezine ürün kabulü yapılan tüm palet girişlerin tutulduğu tablodur. Paletin Dağıtım merkezi içindeki tüm stok hareketleri ayrıca bu tabloda bulunan adres bölümünde tutulmaktadır. Çift taraflı kontrol için oldukça önemlidir.

Tablo 6.3: Ürün Adres İlişki Tablosu

URUN_ADRES
URUN_NO
ADRES
PALET1
PALET2
ISYERI_TURU

Amacı: Ürünlere ait şarj/toplama adreslerinin tutulduğu tablodur.

Ek A.2 Adres Bilgilerinin İşlenmesi

```
INSERT INTO Urun_adres_stok
VALUES
('3894','C01A01A',null,null,null,to_date('14/05/2015','DD/MM/RRRR'),null,null,null);

INSERT INTO Urun_adres_stok
   VALUES
('3894','C01A01B',null,null,null,to_date('14/05/2015','DD/MM/RRRR'),null,null,null);

INSERT INTO Urun_adres_stok
   VALUES
('3894','C01A01C',null,null,null,to_date('14/05/2015','DD/MM/RRRR'),null,null,null);
```

Ek A.3 Palet Bilgilerinin İşlenmesi

```
INSERT INTO Palet_Giris
   VALUES

('PLT00001313082','06035126','12','40','480',to_date('03/08/2015','DD/MM/RRRR'),'56
7','1','977','76541',to_date('03/08/2015','DD/MM/RRRR'),to_date('01/02/2018','DD/MM
/RRRR'),'GIRISTE',null,'39','479',null,null);

INSERT INTO Palet_Giris
   VALUES

(PBARCODE,MAL_NO,SIM,ST,MIKTAR,ISLEM_TRH,KULLANICI_NO,FIS_TUR
,YER_KODU1,FIS_NO,FIS_TRH,SKT,ADRES,PARTI_NO,BAKIYE_ST,BAKIYE_
MIKTAR,BITTI,SKT_ENV)                                    values
('PLT00001313083','06035126','12','40','480',to_date('03/08/2015','DD/MM/RRRR'),'56
7','1','977','76541',to_date('03/08/2015','DD/MM/RRRR'),to_date('01/02/2018','DD/MM
/RRRR'),'GIRISTE',null,'40','480',null,null);
```

Kk A.4 Adres Bazlı Stok Hareket Fonksiyonları

Paletli ürünlerin dağıtım merkezi içindeki hareketleri oracle veri tabanında ABS işlemleri adı altında bir stored package içinde geliştirilmiştir. Stored procedure içinde prosedürler sırasıyla incelenirse;

Procedure YERDEN_RAFA_KOY
Ürün paletinin dağıtım merkezine kabulü yapıldıktan sonra adresi 'GIRISTE' olarak düzenlenir. YERDEN_RAFA_KOY prosedürü uygulama üzerinden paletin yerden alınıp bir raf adresine yerleştirilmesi esnasında kullanılır.

Procedure RAFTAN_YERE_INDIR_PALET
Rafa yerleştirilen paletlerin bir nedenden dolayı yere indirilmesi gerekiyorsa, uygulama üzerinden palet numarası okutulmuş ise palet üzerinden raf bilgisine erişilir ve yere indir komutu geldiğinde RAFTAN_YERE_INDIR_PALET prosedürü çağrılır.

Procedure RAFTAN_YERE_INDIR_RAF
Rafa yerleştirilen paletlerin bir nedenden dolayı yere indirilmesi gerekiyorsa, uygulama üzerinden raf bilgisi okutulmuş ise raf üzerinden palete bilgisine ulaşılır ve yere indir komutu geldiğinde RAFTAN_YERE_INDIR_PALET prosedürü çağrılır.

Procedure RAFLAR_ARASI_HAREKET
Rafta bulunan bir ürün paletini farklı bir rafa taşınması gerekiyorsa, uygulama üzerinden iki adres bilgisi girişi yapıldıktan sonra taşı komutu girildiğinde RAFLAR_ARASI_HAREKET prosedürü çağrılır.

Procedure PALET_TARIHCE_KAYIT
Tüm palet hareketlerinin tarihçelerinin tutulmasını sağlayan prosedür.