YAZILIM GELİŞTİRME SÜRECİNDE DEĞER ODAKLI İYİLEŞTİRME

Şeyda SERDARASAN[1*], Ebru ERTEK[2]

[1]İstanbul Teknik Üniversitesi, İşletme Fakültesi, Endüstri Mühendisliği Bölümü, İstanbul
ORCID No: https://orcid.org/0000-0001-9933-0998

[2]İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Endüstri Mühendisliği Programı, İstanbul
ORCID No: https://orcid.org/0000-0002-3052-2956

Anahtar Kelimeler

Yazılım geliştirme, değer, israf, değer akış haritalama, iki adımlı kümeleme

Öz

Yazılım geliştirme süreci, bir yazılım ürününün geliştirilmesine uygulanan süreçtir. Yazılım geliştirme sürecinin bir yandan müşteri memnuniyetini sağlayacak kaliteli bir yazılım ürününü son kullanıcıya daha etkin bir iletişim ve görünürlük ile sunması, bir yandan da toplam tedarik süresini azaltması ve kaynak kullanım verimliliğini artırması istenir. Bu çalışmada bu amaçla yazılım ürünü ve yazılım geliştirme süreci değer odaklı olarak ele alınmış ve yalın ilkeler çerçevesinde iyileştirme adımları tanımlanmıştır. Çalışma kapsamında yapılan uygulamada, yazılım geliştirme süreçleri için değer tanımı yapılmış ve bu değer tanımından yola çıkarak projelerin sınıflandırılabilmesi için kriterler belirlenmiş ve projeler iki adımlı kümeleme analizi ile gruplanmıştır. Her bir proje grubu için mevcut durum değer akış haritası çizilmiştir. Bu haritalardan yola çıkarak israflar tespit edilmiş, çevrim süresini kısaltacak ve süreci iyileştirecek öneriler sunulmuştur. Genel olarak yazılım projelerinde israfı gereksiz kodlar, iyi tanımlanmamış ihtiyaçlar, bürokrasi ve yavaş iç iletişim oluşturmaktadır. Müşteri ile sık sık iletişim, prototip oluşturma, kaynak sorununu anlama, durum netleşinceye kadar bağlayıcı kararlar alınmaması israfları azaltan uygulamalar olarak karşımıza çıkmaktadır.

VALUE ORIENTED IMPROVEMENT FOR SOFTWARE DEVELOPMENT PROCESS

Keywords

Software development, value, waste, value stream mapping, two step cluster analysis.

Abstract

The software development process is used to regulate and control the development process of a software product. A successful software development process not only delivers a quality software product that improves customer satisfaction with more effective communication and visibility to the end user, but also simultaneously reduces the total lead time and increases resource efficiency. With this in mind, in this study, the software product and software development process are discussed from a value-oriented perspective and implementation steps for improvement are defined within the framework of lean principles. The proposed steps were applied in an IT department: the value definition was made for software development processes and based on this definition, the criteria for the classification of projects were determined. The projects were grouped using two-step cluster analysis. A current state value stream map is drawn for each project group. Based on these maps, wastes were determined, suggestions were made to shorten the cycle time and improve processes. In general, software project wastes are unnecessary codes, poorly defined needs, bureaucracy and slow internal communication. Frequent communication with the customer, creating prototypes, understanding the resource problem, not making binding decisions until the facts are determined are practices that would reduce waste.

Araştırma Makalesi		Research Article	
Başvuru Tarihi	: 13.10.2020	Submission Date	: 13.10.2020
Kabul Tarihi	: 11.02.2021	Accepted Date	: 11.02.2021

*Sorumlu yazar; e-posta: serdars@itu.edu.tr

1. Giriş

Yazılım geliştirme süreci veya yaşam döngüsü, bir yazılım ürününün geliştirilmesinde uygulanır. Yazılım geliştirme süreç metodolojisi, yazılım ve bilgi sistemleri geliştirme sürecini planlamak, yürütmek ve kontrol etmek için kullanılan çerçeveyi tanımlar. Zaman içinde her birinin kendine özgü güçlü ve zayıf yönleri olan çok sayıda farklı metodoloji ortaya konmuştur ve bunlar gittikçe artan sayıda işletme tarafından yazılım geliştirme süreçlerinde uygulanmaktadır. Stephens (2015) yazılım süreç metodolojilerini kestirimci (ing. predictive) (örn. Şelale, V-Modeli), yinelemeli (ing. iterative) (örn. Spiral, Cleanroom) ve çevik (ing. agile) (örn. Scrum, XP, Lean) olmak üzere üç başlık altında sınıflandırır. Süreç geliştirme metodolojilerindeki çeşitlilik, tek bir süreç geliştirme metodolojisinin her türlü yazılım geliştirme projesinde ve organizasyonunda kullanıma uygun olmamasından da kaynaklanmaktadır. Mevcut metodolojilerin her biri farklı teknik, organizasyonel, proje ve ekip yapılarını dikkate alarak, belirli proje türlerine uygun biçimde şekillenmiştir. Sık kullanılan yaklaşımların arasında şelale (ing. Waterfall), çevik tümleşik süreç (ing. Agile unified process), Scrum, Test odaklı geliştirme (ing. Test-driven development), Yalın yazılım geliştirme (ing. Lean software development) sıralanabilir (Vijayasarathy ve Butler, 2016).

Bir yazılım geliştirme süreci ihtiyaçların belirlenmesi, tasarım, test, uygulamaya alma (taşıma/teslimat) ve bakım adımlarını içerir. Süreç metodolojilerini birbirinden ayıran temel fark bu adımların nasıl ve hangi sırayla ele alındığı konusundadır. Sık kullanılan metodolojilerin işleyişi ve avantajları kısaca aşağıdaki gibi özetlenebilir.

Şelale metodolojisi, diğer deyişle klasik yazılım geliştirme yaklaşımı, temelde bir süreç adımı bitmeden diğer adıma başlanmamasını esas alır. Bir adım tamamlandıktan sonra tekrar o adıma geri dönüş nadir gerçekleşir. Bu nedenle tüm adımların olabildiğince plana uygun işlemesi gerekir. İhtiyaçların net belirlenebildiği, geliştirme süreci içindeki değişkenliğin ve riskin düşük olduğu ve hızın öncelikli olmadığı durumlarda uygulanması önerilir (Stephens, 2015; Pressman ve Maxim, 2020).

Spiral metodolojisi, Boehm (1988) tarafından önerilen, prototiplemenin yinelemeli yapısını şelale metodolojisinin kontrollü ve sistematik yönleriyle birleştiren iteratif bir yazılım süreç yaklaşımıdır. Yazılımın aşamalı olarak daha eksiksiz sürümlerinin hızlı bir şekilde geliştirilmesine olanak verir (Pressman ve Maxim, 2020). Risk analizine vurgu yapar ve değişime ayak uydurma açısından şelaleden daha etkindir. Büyük ölçekli projelerde daha başarılıdır (Stephens, 2015).

Scrum, günümüzde giderek popüler hale gelen Çevik metodolojilerdendir. Scrum (ismini bir ragbi sporundaki bir terimden alır), 1990'ların başında Jeff Sutherland ve geliştirme ekibi tarafından tasarlanan bir yazılım geliştirme yöntemidir (Pressman ve Maxim, 2020). Scrum ilkeleri çevik manifesto (agilemanifesto.org, 2001) ile uyumludur ve yazılım yaşam döngüsü adımlarına rehberlik eder. Bir Scrum projesi sprint adı verilen bir süreç deseni içinde gerçekleşir. Proje için gerekli olan sprint sayısı ürün karmaşıklığına ve büyüklüğüne bağlı olarak değişiklik gösterir. Bir sprint içinde yürütülen faaliyet Scrum ekibi tarafından gerçek zamanlı olarak tanımlanır ve iteratif olarak yürütülür ve her sprint sonunda uygulama canlıya alınabilir. Böylece bir özellik geliştirilirken herhangi bir sorun meydana gelirse bunun projenin kalanı üzerindeki etkisi azaltılmış olur ve risk dağıtılır (Elibol ve Selçukcan Erol, 2017). Scrum zaman kısıtı, ihtiyaçlardaki değişkenlik, ve kritiklik gibi koşullarda iyi sonuçlar vermektedir (Pressman ve Maxim, 2020).

Yalın yazılım geliştirme bir diğer çevik metodolojidir. Yazılım geliştirmede yalın yaklaşım, yalın düşünce prensiplerinin yazılım mühendisliğine uyarlanmasını temel alır (Poppendieck ve Poppendieck, 2003). Bu yaklaşım da özelliklerin küçük parçalara bölünmesi ve geliştirmenin iteratif olarak yürütülmesi konusunda Scrum ile benzerlik gösterir. Yalın metodoloji, değer yaratan gereksinimleri tanımlamaya ve her bir faaliyetin katma değer yaratan (gerekli) bileşenlerden oluşturulmasına odaklanır. Yalın metodolojinin hedefi israfı engellemektir, bu sebeple gerekli olmayan hiçbir özellik geliştirilmez. Çevik ve klasik projelerin zaman, bütçe ve müşteri memnuniyeti hedeflerini sağlama başarılarını karşılaştıran CHAOS raporu (www.standishgroup.com) verilerine göre çevik yaklaşımın başarı oranlarının genel anlamda daha yüksek olduğu yorumu yapılabilir.

Yazılım geliştirme sürecinde kullanılacak süreç metodolojisi seçim kararı yazılım projesinin ölçek ve ihtiyaçları, yazılımı gerçekleştirecek ekibin boyutu, hedefleri, bilgi ve beceri düzeylerinin

yanısıra maliyet, müşteri ve yazılım şirketi tercihleri gibi diğer faktörlere de bağlıdır (Khan ve Beg, 2013; Papadopoulos, 2015)

Bu çalışmanın amacı, müşteri memnuniyetini sağlayacak kaliteli bir yazılım ürününü son kullanıcıya daha etkin bir iletişim ve görünürlük ile sunan, toplam tedarik süresini azaltan, ve kaynak kullanım verimliliğini artıran bir yöntem önerisi ortaya koymaktır. Bu amaçla yalın prensipler çerçevesinde yazılım ürünü ve yazılım geliştirme sürecini değer odaklı olarak ele alan ve yazılım geliştirme sürecini iyileştirmeye yönelik bir yöntem önerilmiş ve bir işletmedeki uygulaması aktarılmıştır.

Çalışmanın akışı şu şekildedir: ikinci bölümde değerin tanımı, değerin yazılım projeleri için tanımı, yazılım geliştirme süreçlerinde yalın düşünme ve israfın yazılım projeleri için tanımı ile ilgili yazın taraması sunulmuştur. Ardından üçüncü bölümde önerilen yöntemin adımları ortaya konmuştur. Dördüncü bölümde kümeleme yöntemi ile yazılım projelerinin sınıflandırılmasını, mevcut durum değer akış haritaları çizilmesini ve iyileştirme önerilerini içeren uygulama aktarılmıştır. Sonuç bölümünde çalışmanın akademik ve yönetsel etkileri tartışılmış, çalışmadaki kısıtlardan bahsedilmiş ve gelecek çalışmaları için önerilerde bulunulmuştur.

2. Bilimsel Yazın Taraması

2.1 Değer tanımı

Değer, kabaca müşterinin ödemeye razı olduğu herşey olarak tanımlanır. Literatürde müşteri değerinin tanımları incelendiğine değerin iki genel anlamı ortaya çıkar: değişim/mübadele değeri ve kullanım değeri (Vargo, Maglio ve Akaka 2008). Mübadele değeri, bir ürün karşılığında insanların ödemeye istekli oldukları ekonomik değer olarak tanımlanabilir (Vargo ve Lusch 2004). Öte yandan kullanım değeri, kullanıcının ürün/hizmet deneyimini değerlendirmesinin sonucudur (Sandström, Edvardsson, Kristensson ve Magnusson 2008).

Tablo 1
Paydaşlara göre değer tanımları, Haksever ve Chaganti (2004)'den uyarlanmıştır

Paydaş Grubu	Değer Boyutu	Yaratılan Değer
Hissedar	Finansal	Kar, gelir, hisse senedi değerlemesi, karlı ve finansal açıdan stabil bir şirket
	Finansal olmayan	Güvenilir gelir kaynağı, özerklik, iyi bir imaja sahip şirkete sahip olmanın gururu
	Zaman	Uzun vadede finansal güvenlik, yeni ürünlere, ileri teknolojiye, kaliteye ve insan kaynağına yapılan yatırımlar, tedarikçilerle uzun vadeli stratejik ortaklıklar, şirketin uzun vadede varlığını sürdürmesi
Çalışan	Finansal	Maaş, ek ödemeler, sağlık sigortası, emeklilik planı, ücretli izin süresi; iyi yönetilen kar eden ve finansal açıdan stabil bir şirkette çalışmak
	Finansal olmayan	Güven, işbirlikçi bir iş çevresi, eğitim, iş güvenliği, yeni fırsatlar, esnek iş kuralları; yönetimin çalışana duyduğu saygı, insan kaynaklarına yapılan yatırım, terfi imkanı
	Zaman	Uzun vadede iş güvenliği, gelecek kariyer olanakları, bağlılığı ödüllendirme
Müşteri	Finansal	Rekabetçi fiyata iyi tasarlanmış, kaliteli ürünler. Güvenilir, az bakım gerektiren, düşük işletme maliyetine sahip ürünler.
	Finansal olmayan	Söz verildiği gibi çalışan, kurulumu ve kullanımı kolay ürünler. Nazik, güvenilir müşteri hizmetleri. Ürünün kullanımı, bakımı ve onarımı hakkında doğru ve yeterli bilgi
	Zaman	Ürünün sağladığı zaman tasarrufu, uzun süreli fayda, söz verilen zamanda teslimat. Arıza durumunda hızlı müdahale, servis hızı, arızalı parçaların hızlı değişimi.

Literatürdeki çalışmalar gözden geçirildiğinde, değer tanımı çalışmalar arasında farklılık göstermekte ve genelde net olarak tanımlanmamaktadır. Yazılım ile ilişkili literatürde ağırlıklı olarak değerin iş değeri olarak da yorumlanan kullanım değeri anlamının kullandığı söylenebilir. Levina ve Ross (2003) değeri "müşterilerine sözleşme maliyetlerinden ve risklerinden daha ağır basan ekonomik ve kullanım faydaları sağlama" olarak tanımlar; bu da kullanım değeri anlamıyla uyumludur. Değer yaratma sürecini anlamak için farklı bakış açılarını dikkate almaya ihtiyaç vardır (Mele ve Polese 2011). Bunlar, müşteri, hizmet sağlayıcı ve tedarikçiler gibi tipik bir hizmet ağına dahil olan taraflardır. Haksever ve Chaganti (2004)'nin değer tanımı bu kapsamla uyumludur. Haksever ve Chaganti (2004) değeri, hissedar, çalışan ve müşteri için finansal, finansal olmayan ve zaman boyutlarında Tablo 1'de görüldüğü gibi tanımlamışlardır. Hissedarların şirkette finansal payları vardır ve yatırımların değer kazanmasını ya da gelir getirmesini beklerler. Çalışanlar, geçimleri için aldıkları ücretlere veya maaşlara güvenirler ve aynı zamanda terfi, sosyal imkanlar gibi ödüllerden faydalanırlar. Müşteriler, satın aldıkları ürünlerden belirli faydalar beklemektedir ve genel olarak ihtiyaçları makul bir fiyata karşılandığı sürece şirkete sadık kalırlar. (Haksever ve Chaganti, 2004)

2.2 Değerin Yazılım Projeleri İçin Tanımı

Değere odaklanma, yazılım yoğun ürün geliştirme için kritik başarı faktörlerini araştıran, başarısız yazılım projelerinden başarılı olanları irdeleyen çalışmaların çoğunda ele alınan bir konudur (Boehm, 2006). Bununla birlikte, değerin bir kavram olarak ortak net bir tanımının olması durumu sınırlıdır (Dingsoyr, Nerur, Balijepally ve Moe, 2012; Dyba ve Dingsoyr, 2008; Racheva, Daneva ve Sikkel, 2009).

Poppendieck (2011)'e göre değer kavramı ile yalın prensipler arasında güçlü bir bağ vardır ve yalın yazılım geliştirmenin ilk prensibi, 'şirkete veya müşterilerine bir değer katmadığı sürece her faaliyet kaynakları boşa harcar' şeklindedir.

Yazılım yoğun ürün ve hizmetler geliştiren bir şirketin amacı, belirli bir yatırım için değer yaratma özelliğini en üst düzeye çıkarmaktır. Bunu gerçekleştirmek için neyin değer olarak kabul edildiğini, neyin değer yarattığını, ve değer yaratan stratejilerin ne olduğunu anlamak gerekir (Aurum ve Wohlin, 2007). Literatürde, yazılım geliştirme süreci için tanımlanan değer boyutları aşağıda sıralanmıştır.

Doğruluk: Bir yazılım geliştirilmeden önce analist ve proje paydaşları bir araya gelerek gereksinimler doğrultusunda spefikasyonlar tanımlanır. Ortaya konan yazılımın tanımlanan spefikasyona sadık kalması bir değer boyutudur. Bu, yazılımın doğru çalışıp çalışmadığının da ölçütüdür. (Ebert ve Dumke, 2007)

Güvenilirlik: Yazılım projesi uygulamaya geçtikten sonraki süreç içerisinde, yani canlı ortamda, ne sıklıkla bir hata oluştuğudur. (Ebert ve Dumke, 2007; Alahyari, Svensson ve Gorschek, 2017)

Veri Bütünlüğü ve Güvenliği: Yazılımlar genellikle bir veri kitlesini saklama sorumluluğuna sahiptir. Veri bütünlüğü, bilginin eksiksiz ve bozulmamış olmasını ve hiçbir şekilde değiştirilmemiş olmasını sağlama becerisidir. Veri güvenliği ise, bilgi varlıklarının yalnızca bunlara erişme yetkisi olan kişiler tarafından kullanılabilir olmasını sağlama becerisidir. (Futcher ve von Solms 2008)

Teslim zamanı: Yazılımın müşterisi her zaman ürünün teslimatının zamanında yapılmasını bekler. Müşteri, kendi taahhütlerini ve ihtiyaçlarını yerine getirmek için yazılıma söz verilen tarihte ihtiyaç duyar. Yazılımın son teslim tarihinde teslim edilememesi durumunda, müşteri ürünü teslim almaktan vazgeçerek, başka bir firmaya yönelebilir. Bu nedenle yazılımın geç teslim edilmesindense hatalı şekilde teslim edilmesi tercih edilebilmektedir. (Janzen ve Saiedian, 2005; Alahyari ve diğ. 2017)

Minimum maliyet: Müşteri, istenen işlevselliği ve kaliteyi elde edecek en düşük fiyatı ödemek ister. (Janzen ve Saiedian, 2005; Alahyari ve diğ. 2017)

Kullanım kolaylığı: Kullanıcıya anlaşılır bir arayüz sunması ve uygulamanın kullanılabilirliğinin yüksek olması (Alahyari ve diğ. 2017)

Geliştirmeye açık olması: Yazılım ürününün gelecek ihtiyaçları karşılayabilecek eklentilere açık olmasını anlatır. bu, bir yazılımı geliştirme becerisinin ve eklentileri uygulamak için gereken çaba düzeyinin bir ölçüsüdür. geliştirmeler, yeni işlevlerin eklenmesi veya mevcut işlevselliğin değiştirilmesi yoluyla olabilir ve mevcut sistemin işlevlerini bozmadan geliştirme sağlanır. (Cooke 2016)

Bu çalışmada, Haksever ve Chaganti (2004)'ün ortaya koyduğu gibi, değer tanımı yapılırken sadece müşteri değil diğer paydaşlar da dikkate alınmıştır.

Müşteri açısından teslim zamanı, doğruluk, kullanım kolaylığı, geliştirmeye açık olması, minimum maliyet boyutları ön plana çıkarken, çalışan için veri bütünlüğü ve güvenliği, güvenilirlik ve zaman açısından değerlendirildiğinde minimum maliyet boyutları tanımlanabilir. Hissedar açısından ise teslim zamanı, minimum maliyet, doğruluk boyutları değeri tanımlamaktadır. Değer boyutlarının ortaya konması yazılım projelerinin değerinin ölçülebilir ve gözlemlenebilir ölçütlerle ilişkilendirilmesinde önem taşımaktadır.

2.3 Yazılım Geliştirme Süreçlerinde Değer Akış Haritalama

Yadav, Mittal ve Jain (2019) yazılım geliştirme projelerinde yalın ilkelerin benimsenme düzeyi hakkındaki çalışmalarında yalın ilkeleri (değer, değer akışı, akış, çekme ve mükemmellik) ve uygulama araçlarını incelemişlerdir. Yalın yazılım projelerinin performans, döküm, teslim süresi, müşteri memnuniyeti, maliyet ve iş değeri üzerindeki etkisi analiz edilmiştir (Yadav ve diğ., 2019). Kullanılması önerilen yalın araçlardan biri de Değer akış haritalamadır. Değer akış haritalama sırasında süreçteki her bir faaliyet analiz edilir ve bu faaliyetlerin sürece değer katıp katmadığı sorgulanır.

Yazılım geliştirme sürecine değer akış haritalama uygulanması konusunda az sayıda yayın bulunmaktadır. Scopus veri tabanında Ekim 2020'de (TITLE-ABS-KEY ("software development" OR "software product") AND TITLE-ABS-KEY ("value stream map" OR "value stream mapping"))" arama terimleri ile yapılan arama sonucunda 2013 ile 2020 yılları arasında yayınlanmış sadece 10 çalışma (makale ve bildiri) listelenmiştir. Bu çalışmalardan bazıları şöyle sıralanabilir:

Mujtaba, Feldt ve Petersen (2010) yazılım özelleştirme sürecindeki israf kaynakları sorunları ortaya koymak için değer akış haritalama kullanmışlardır.

Staron, Meding ve Caiman (2013) ölçüm sistemleri tarafından sağlanan bilgilerin eksiksizliğini değerlendirmek için değer akış haritalamaya dayalı bir yöntem sunmuşlardır. Bu yöntem, bir yazılım geliştirme ortamında paydaşlara yazılım kalitesiyle ilgili gelecek sorunlar hakkında erken bir uyarı sistemi sağlamak için uygulanmıştır.

Anand, Chandrashekar ve Narayanamurthy (2014) lojistik tedarikçilerine tedarik zinciri yazılım çözümleri sağlayan bir Hint yazılım firmasının iş sürecini yeniden yapılandırmak için farklı yalın araçlar sunarak değer akış haritalama uygulaması ile çeşitli israfları tespit etmişlerdir.

Ali, Petersen ve Schneider (2015) değer akış haritalamayı yazılım süreci simülasyon modellemesi ile birleştirerek Ericsson AB'nin İsveç'teki iki ürünü için kullanmışlardır. Yazarların 2016 yılında yürüttükleri çalışmada ise İsveç AB Ericsson'daki büyük bir ürün için değer akış haritalama, sistematik olarak sorunların ortaya çıkarılmasında ve çözümlerinin karakterize edilmesinde başarıyla kullanılmıştır (Ali, Petersen ve Schneider, 2016).

Aktaş, Altunel, Elalmış, Nişancı ve Yılmaz (2018) yazılım sorunlarının takibi amacıyla kullanılan olay kaydı yönetim sistemlerini ele almışlardır. Çalışmada, Türkiye'de bankacılık sektöründe hizmet veren bir yazılım şirketinde olay kaydı çözüm süreci "değer akışı haritalama" ile incelenmiş, ilgili veriler ölçülerek iyileştirme amacıyla analiz edilmiş ve elde edilen iyileştirmeler paylaşılmıştır.

Özcan-Top ve McCaffery (2018) MDevSPICE tabanlı süreç değerlendirme yöntemini, iyileştirme ihtiyaçlarının önceliklendirilmesi için değer akış haritalama ve süreç iyileştirme için KATA tekniği ile birleştirmiştir. Önerilen yaklaşım ile çevik yöntemler yasal yönetmeliklerin gerekliliklerini yerine getirerek tıbbi cihaz yazılımı geliştirme sürecine entegre edilmiştir.

Murphy ve Kersten (2020), değer akışının yazılım geliştirmeye nasıl uygulandığını ve değere daha fazla odaklanabilmek için özelliklerin daha sistematik olarak ele alınmasının ne kadar önemli olduğunu DevOps uygulaması üzerinden açıklamışlardır.

İncelenen tüm çalışmalarda ortak olarak yazılım geliştirme alanında değer akış haritalama yöntemi kullanılarak israflar belirlenmiştir ve israfların ortadan kaldırılabilmesi için analizler ile süreç iyileştirmeleri yapılmıştır. Bu çalışmada da benzer şekilde yazılım geliştirme süreci için değer akış haritalama metodu kullanılmış ve israflar belirlenerek süreçte çevrim süresini kısaltacak iyileştirme önerileri sunulmuştur. Diğer çalışmalardan farklı olarak bu çalışmada projeler değer tanımından yola çıkarak iki adımlı kümeleme analizi ile gruplanmıştır.

2.4 Yazılım Geliştirme Süreçlerinde İsrafın Tanımı

Yalın ilkelerin başında değerin tanımlanması ve değer akışının haritalanarak israfların belirlenmesi gelmektedir. Ohno (1988) yedi israf kaynağından bahsetmektedir: Fazla üretim, taşımalar, beklemeler, yeniden işleme, stoklar, hareket ve hatalar. Bauch (2004) ürün geliştirme kapsamında bu israflara yeniden keşif, sistem disiplin yoksunluğu ve bilgi teknolojileri kaynaklarının sınırlı olmasını da eklemiştir. Uygulama alanından bağımsız olarak her tür sistemde bu sayılan başlıklar altında değer katmayan faaliyetler belirlenebilir.

Yedi temel yalın üretim israfı, yazılım geliştirme projelerine aşağıdaki şekilde uyarlanmıştır (Poppendieck ve Poppendieck 2003; Sedano, Ralph ve Péraire, 2017; Alahyari, Gorschek ve Svensson 2019).

1. Kısmen yapılmış iş (partially done work): Entegre edilmeyen, test edilmeyen ve eskimiş çalışmalar anlamına gelmektedir. Burada asıl sorun özelliğin işe yarayıp yaramayacağını bilememektir.
2. Ek süreçler (extra-processes): Gerekli olmayan veya bir işlemin sonuna kadar ertelenebilen süreçlerdir.
3. Ek özellikler (extra features): Sisteme ekstra özellik eklemek zararsız gibi görünebilir hatta iyi bir fikirmiş gibi gelebilir, ancak gerekenden fazla özellik veya yanlış özellik israf olarak kabul edilir.
4. Görev değiştirme (task switching): Görev değişikliği yapmak yazılım geliştirme sürecindeki çalışanların görev değişikliği yapmaları her geçişte düşünceleri yeni görev akışı için sıfırdan tazelemek anlamına gelmektedir.
5. Bekleme (waiting): Bir yazılım geliştirme sürecindeki en önemli israflardan biridir. Projenin başlamasındaki gecikmeler, tekrar gözden geçirmeler, onay bekleme, test ve canlı ortama geçişteki gecikmelerden oluşmaktadır. İş akışında bekleme, çoklu görevlerden kaynaklı yükleri de beraberinde getirir.
6. Hareket (motion): Yazılım geliştirme sürecindeki çalışanların bir sorun veya gereksinim için harekete geçmesi ile ilgilidir. Genelde yeniden aynı iş üzerinde çalışma, iş yükünün yanlış yönetilmesi şeklinde karşımıza çıkar.
7. Hatalar (defects): Süreçteki en gözle görülebilen israf kaynağıdır. Hatanın israf miktarı hatanın etkisine ve tespit edildiği süreye bağlı olarak ortaya çıkmaktadır.

Alahyari ve diğ., (2019) çalışmalarında bu israfları önceliklendirmiş ve aralarındaki ilişkileri ortaya koymuşlardır. Buna göre görev değiştirme, hatalar ve bekleme yazılım projelerinde karşılaşılan israfların başında gelmektedir.

3. Yöntem

Bu çalışmanın yöntemsel temelini oluşturan yalın ilkeler Womack ve Jones (1996) tarafından değeri tanımla (ürün - müşteri ve süreç odaklı); değer akışını haritala ve değer yaratmayan adımları (israfları) belirle; israfı azalt, akışı sağla; değer odaklı yeni akış önerisi geliştir ve sürekli mükemmeli ara şeklinde ortaya konmuştur. Bu çalışmada yalın ilkeler çerçevesinde yazılım projelerinin değer odaklı analizi ve iyileştirilmesi için izlenmesi önerilen adımlar şöyle tanımlamıştır:

1. Değerin tanımlanması: Müşteri, ürünler, işletme süreçleri ve diğer paydaşlar dikkate alınarak yazılım projeleri için değer tanımının yapılması ve değer boyutlarının (kriterlerinin) belirlenmesi. Bu aşamada literatürde yazılım geliştirme ile ilgili değer tanımlarından yola çıkan ve yazılım sağlayıcının mevcut durum ve beklentileri de dikkate alan değer tanımları yapılmalıdır. Tanımlanan değerin ölçülebilir ve/veya gözlemlenebilir boyutlara dönüştürülmesi ve bunlara ait verinin erişilebilir olması önem taşımaktadır.

2. Projelerin gruplanması: Projenin doğası gereği bir defaya mahsus olması yazılım projeleri için de geçerlidir. Bu da yalın ilkelerde önerildiği gibi ürünleri ürün ailesi bazında ele alma konusuna yeni bir yaklaşım gerektirmektedir. Her ne kadar projeler özgün olsalar da süreç, müşteri beklentisi vb. konularda ortak özellikler taşımaktadırlar. Bu çalışmada, değer odaklı bir yaklaşımla benzer özellikli projelerin kümelenmesi esas alınmıştır. Bu amaçla tanımlanan değer boyutları kullanılarak uygun bir kümeleme tekniği ile projelerin kümelenmesi önerilmiştir. Belirlenen boyutların bu çalışmada ele alınan uygulamada olduğu gibi hem metrik hem kategorik değişkenler içermesi halinde iki adımlı kümeleme analizinin kullanımı önerilebilir. Kümeleme analizine girdi sağlayacak veriler yazılım geliştirme araçları (örn Jira -

www.atlassian.com/software/jira) üzerinden sağlanabilir.

3. Proje mevcut değer akış haritalarının çizilmesi: Değer akışı, bir müşterinin talebinden bu talebin yerine getirilmesine kadar olan iş akışıdır. Değer akışı haritalama yöntemi, değer akışını görselleştirmek için kullanılan yalın planlama aracıdır ve bir süreçte yer alan israfları belirlemek için ana yaklaşım olarak kabul edilmektedir (Rother ve Shook, 2003; Poppendieck ve Poppendieck 2003). Ortak bir kümede yer alan projelerin işlem adımları, işlem ve bekleme süreleri belirlenerek mevcut durumun akışı ortaya konur.

4. İsrafların belirlenmesi, iyileştirme önerileri. Yazılım geliştirmede sık karşılaşılan israflar Poppendieck ve Poppendieck (2003), Alahyari ve diğ., (2019) tarafından listelenmiştir. Mevcut süreç, değer tanımları çerçevesinde değer yaratan ve yaratmayan aktiviteler olarak irdelenip israflar ortaya konur. İsrafların ortadan kaldırılmasına veya azaltılmasına yönelik iyileştirme önerileri sunulur.

5. Sürecin iyileştirilmesi ve gelecek durum haritasının çizilmesi. Önerilen iyileştirme önerilerinin uygulamaya alınması ile ulaşılacak olan gelecek durumun haritası çizilir ve yeni işlem süreleri hesaplanır (Rother ve Shook, 2003).

Bu çalışmada araştırma ve yayın etiğine uyulmuştur.

4. Uygulama

Bu çalışmada İstanbul'daki bir finans firmasının Bilgi Teknolojileri departmanında yürütülen yazılım geliştirme projeleri ele alınmıştır. Yazılım geliştirme projelerinde 32 analist ve 29 yazılımcı olmak üzere toplam 61 personel görev almakta ve yazılım geliştirme metodu olarak şelale yaklaşımı kullanılmaktadır. Departmanın yürüttüğü 154 yazılım geliştirme projesi ele alınmıştır.

4.1 Değer Tanımı ve Proje değerlendirme Kriterlerinin Belirlenmesi

Bilimsel Yazın Taraması başlığında aktarılan değer tanımlarından ve boyutlardan yola çıkarak yazılım projeleri için Tablo 2'de yer alan kriterler ortaya konmuştur.

Tablo 2
Projeleri kümelemede kullanılan kriterler ve özellikleri

Kriter	Açıklama	Veri tipi	Kodlama
Maliyet	Tahmin edilen süre	Nümerik (gün)	-
Efor	Harcanan efor	Nümerik (gün)	-
Öncelik	Projenin önceliği var mı?	Kategorik	0-1
Amaç	Projenin amacı	Kategorik	1-5
Kaynak	Projede çalışan kişi sayısı	Nümerik	-
EkTalep	Ek talep sayısı	Nümerik	-
Güncelleme	Güncelleme sayısı	Nümerik	-
Tecrübe	Projede çalışanların tecrübe yılı toplamı	Nümerik (yıl)	-

Maliyet: Toplam proje süresini ifade etmektedir. Bir iş için harcanması planlanan minimum süre maliyet olarak nitelendirilmektedir. Müşteriye günlük 8 saatten hesaplanarak gün cinsinden bildirilir.

Efor: İşin tamamlanması için harcanan süreyi ifade etmektedir. Teslim zamanı harcanan çabanın toplamı ile hesaplanabilmektedir. Maliyet tanımında olduğu gibi 1 günün 8 saat olduğu varsayılarak gün cinsinden hesaplanır.

Öncelik: Projenin öncelikli olup olmadığını ifade etmektedir. Örneğin gelir getirici, yasal zorunluluğu olan projeler önceliklidir ve planlama yapılırken diğer projelerden daha önde tutulurlar.

Amaç: Projenin ne amaçla yapıldığını ifade etmektedir. 5 farklı kategoriden oluşmaktadır.

- İyileştirme: Mevcut ekranlar üzerinde düzenlemeler yapılması
- Yeni Ürün: Varolmayan bir projenin yapılması
- Yasal Zorunluluk: Mevzuat gereği zorunlu olması
- İç Geri Besleme: Belirlenen hataları düzenlenmesi
- Raporlama: Rapor oluşturulması

Kaynak: Projede çalışan toplam kişi sayısıdır.

Ek Talep Sayısı: Proje yürütülürken istenen ek taleplerdir. Bu aynı zamanda projenin geliştirmeye açık olduğu anlamına gelmektedir. Sayı olarak ifade edilir. Her yeni istek için bir ek talep kartı açılır.

Güncelleme Sayısı: Proje yürütülürken meydana gelen hatalar için yapılan güncellemelerin sayısıdır. Doğruluk ve güvenilirlik değer boyutları ile ilişkilidir. Tanımlanan spesifikasyon ile ortaya konan ürünün farkı güncelleme sayısı ile orantılıdır, güncellermelerin çokluğu israf olarak tanımlanabilir.

Tecrübe: Projede çalışanların toplam deneyim yılı sayısını ifade etmektedir.

4.2 Yazılım Projelerinin Gruplanması

Çalışmada, değer odaklı bir yaklaşımla benzer özellikli projelerin kümelenmesi esas alınmıştır. Kümeleme, verileri birbirine "benzer" ve birbirinden "farklı" veri noktaları içeren kümeler halinde gruplandırmak için kullanılan bir makine öğrenme tekniğidir. Uygulamada, yazılım projelerini değerlendirmek için hem kategorik hem de metrik veriler bulunduğundan projeleri kümelemek için IBM SPSS İki Adımlı Kümeleme Analizi (TwoStep Cluster Analysis) kullanılmıştır. İki adımlı kümeleme analizi klasik kümeleme analizi yöntemleriyle karşılaştırıldığında, hem sürekli hem de kategorik verilerin analizine olanak sağlar (Ahmad ve Khan 2019).

Correlations

		maliyet	efor	priority	amack	ektalep	guncelleme	kaynak	tecrube
maliyet	Pearson Correlation	1	,051	,334**	-,045	,393**	,284**	,268**	,224**
	Sig. (2-tailed)		,531	,000	,579	,000	,000	,001	,005
	N	154	154	154	154	154	154	154	154
efor	Pearson Correlation	,051	1	,124	-,037	-,071	,069	,073	-,149
	Sig. (2-tailed)	,531		,125	,648	,383	,395	,371	,065
	N	154	154	154	154	154	154	154	154
priority	Pearson Correlation	,334**	,124	1	-,069	,215**	,284**	,176*	,138
	Sig. (2-tailed)	,000	,125		,392	,007	,000	,029	,087
	N	154	154	154	154	154	154	154	154
amack	Pearson Correlation	-,045	-,037	-,069	1	,164*	,116	,028	,154
	Sig. (2-tailed)	,579	,648	,392		,042	,151	,732	,056
	N	154	154	154	154	154	154	154	154
ektalep	Pearson Correlation	,393**	-,071	,215**	,164*	1	,475**	,483**	,413**
	Sig. (2-tailed)	,000	,383	,007	,042		,000	,000	,000
	N	154	154	154	154	154	154	154	154
guncelleme	Pearson Correlation	,284**	,069	,284**	,116	,475**	1	,346**	,351**
	Sig. (2-tailed)	,000	,395	,000	,151	,000		,000	,000
	N	154	154	154	154	154	154	154	154
kaynak	Pearson Correlation	,268**	,073	,176*	,028	,483**	,346**	1	,432**
	Sig. (2-tailed)	,001	,371	,029	,732	,000	,000		,000
	N	154	154	154	154	154	154	154	154
tecrube	Pearson Correlation	,224**	-,149	,138	,154	,413**	,351**	,432**	1
	Sig. (2-tailed)	,005	,065	,087	,056	,000	,000	,000	
	N	154	154	154	154	154	154	154	154

Şekil 1. Kriterler arası korelasyon matrisi

Toplam 154 yazılım projesi için kümeleme analizine girdi sağlayacak veriler Jira üzerinden toplanmıştır. Analizde kullanılan öncelik ve amaç değişkenleri kategorik, maliyet, efor, kaynak, ek talep sayısı,

güncelleme sayısı ve tecrübe yılı toplamı değişkenleri metrik değişkenlerdir. Değişkenler iki adımlı kümeleme analizinin normallik ve multinomiallik varsayımlarını karşılamamasına rağmen, yöntem varsayımların ihlaline karşı duyarsız (robust) olduğu için (Norusis, 2012) analize devam edilmiştir. Bağımsızlık varsayımını test etmek için korelasyon matrisine başvurulmuştur. Sarstedt ve Mooi (2019, s.334) yüksek korelasyonlu ($r \geq .9$) değişkenleri ortadan kaldırmayı veya değiştirmeyi önermişlerdir. Değişkenler arası en güçlü korelasyon Şekil 1'de görüldüğü gibi ek talep ile kaynak arasındaki ilişkidir ($r = .483$), bu değer Sarsedt ve Mooi (2019) tarafından belirlenen eşiğin altında olduğu için analize devam edilmesinde sakınca görülmemiştir. Uygulama sonucunda BIC kriterine göre küme sayısı 4 olarak belirlenmiştir. İki Adımlı Kümeleme analizi sonuçları Tablo 3 ve Tablo 4'te görülmektedir.

Tablo 3.
İki adımlı kümeleme analizi sonuçları

Küme sayısı	Bayes Bilgi Kriteri (BIC)	BIC değişimi[a]	BIC değişim oranı[b]	Uzaklık ölçüleri oranı[c]
1	1194,331			
2	1039,945	-154,386	1,000	1,439
3	958,783	-81,163	0,526	1,758
4	949,523	-9,259	0,060	1,663
5	978,091	28,567	-0,185	1,148
6	1014,033	35,943	-0,233	1,456
7	1065,548	51,514	-0,334	1,033
8	1118,157	52,609	-0,341	1,055
9	1172,476	54,319	-0,352	1,191
10	1231,827	59,351	-0,384	1,015
11	1291,570	59,744	-0,387	1,463
12	1359,501	67,931	-0,440	1,006
13	1427,540	68,039	-0,441	1,183
14	1498,305	70,765	-0,458	1,130
15	1570,779	72,475	-0,469	1,074

a. Bir önceki küme sayısındaki değişim.
b. İki küme arasındaki değişim oranı.
c. Önceki küme sayısı ile mevcut küme sayısına arasındaki uzaklık ölçüleri oranı.

Tablo 4
Projelerin kümelere dağılımı

Küme	N	% of Combined	% of Total
1	30	%19,5	%19,5
2	56	%36,4	%36,4
3	29	%18,8	%18,8
4	39	%25,3	%25,3
Combined	154	%100,0	%100,0
Total	154		%100,0

Tablo 5, tüm projelerin ve her bir yazılım projesi kümesinin profilini özetlemektedir. Metrik değişkenler olan maliyet, efor, kaynak, ek talep, güncelleme sayısı ve tecrübe yılı toplamı değerlerinin ortalamaları ve her bir küme içerisindeki ortalamaları, öncelik ve amaç kategorik

değişkenlerinin de toplam ve kümeler bazında hangi sıklıkla yer aldıkları görülmektedir.

Tablo 5
Kümelerin profilleri

Özellik	Toplam	Küme-1 n=30	Küme-2 n=56	Küme-3 n=29	Küme-4 n=39
Maliyet	15,11	36,37	9,29	12,52	9,06
Efor	14,16	19,74	11,87	10,64	15,75
Kaynak	2,54	3,57	2,39	2,14	2,28
EkTalep	1,53	4,46	0,92	0,69	0,77
Güncelleme	4,50	12,37	2,45	3,90	1,85
Tecrübe	13,54	19,75	12,26	12,09	11,71
Öncelik					
Evet	81	3	56	0	22
Hayır	73	27	0	29	17
Amaç					
Gelir getirici	1	1	0	0	0
Maliyet düşürücü	1	0	0	1	0
Raporlama	21	0	0	0	21
Ürün ve hizmet geliştirme	111	27	56	28	0
Yasal zorunluluk	20	2	0	0	18

Tek tek değişken bazında incelendiğinde yüksek maliyete ve efora sahip projeler Küme-1'de toplanmıştır. Küme-1'de kaynak sayısı da diğer kümelere göre daha yüksektir ancak diğer kümeler de yakın değerde dağılım göstermiştir. Ek talep sayısı, güncelleme sayısı ve tecrübe yılı sayısı yine Küme-1'de en yüksek gözlemlenmiştir. Buradan Küme 1 hakkında şu çıkarımları yapabiliriz: Proje maliyeti ve kaynak gereksinimi yüksek, daha fazla efor gerektiren, ek talep sayısı ve hata miktarı (güncelleme sayısı) yüksek, daha tecrübeli personelin çalıştığı projelerdir.

Küme-2 tamamen öncelikli projelerden ve ürün ve hizmet iyileştirme amaçlı projelerden oluşmaktadır. Ortalama olarak en az efor sarfedilen projeler, önceliği olmayan projelerin yer aldığı Küme-3'te toplanmıştır. Raporlama ve yasal zorunluluk amaçlı projelerin hemen hepsi Küme-4'de bulunmaktadır.

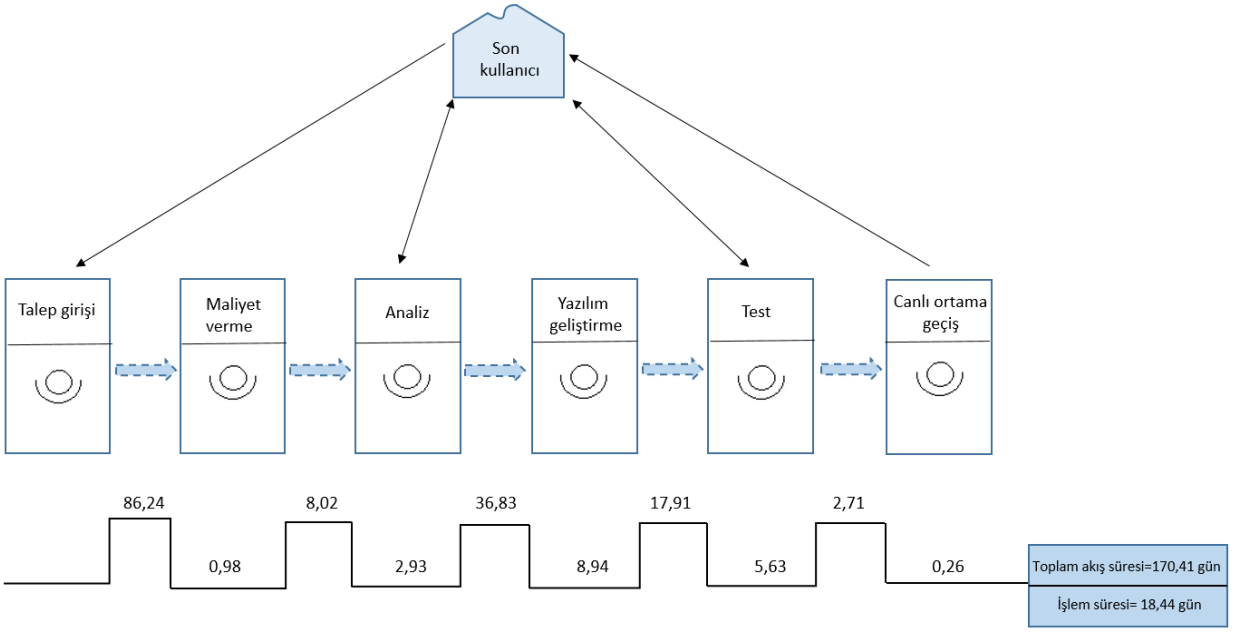
Kümeleme yapılırken değişkenlerin önem sıralamaları amaç, öncelik, maliyet, efor, kaynak, ek talep, güncelleme, tecrübe yılı şeklinde olmuştur. En büyük öneme sahip değişken olan amaç, hissedar için önemli bir kriterdir. Projenin öncelikli olup olmaması müşteri açısından değer olarak tanımlanmaktadır. Maliyet ve efor değişkenleri benzer önem derecelerine sahiptir. Maliyet müşteri ve hissedar açısından önemli iken, efor çalışan için önem ifade etmektedir. Ek talep proje yapılırken ne kadar çok ek istek bulunduğunu ifade etmektedir. Ek talep maliyeti ve eforu doğrudan etkilediği için tüm paydaşlar için önem taşımaktadır, aynı zamanda projenin geliştirmeye ne kadar açık olduğunu da göstermektedir. Bu bakımdan müşteri için değeri yansıtmaktadır. Kaynak, hissedarı doğrudan etkilemektedir. Çalışan açısından da ne kadar fazla kaynak o kadar iş olanağı anlamına geldiği için önemli bir değişkendir. Proje yürütülürken meydana gelen hata sayısı yani güncellemelerin sayısının kümeleme üzerinde etkisi diğer kriterlere göre oldukça azdır. Güncelleme sayısı proje maliyetini arttırdığı için müşteri ve hissedar için önemli bir kriterdir. Hataları düzeltmek için harcanan efor da artacağı için çalışan için de önem taşıyan bir kriterdir. Kümeleme sırasında en az ayırıcı etkisi olan değişken ise tecrübe yılı sayısıdır. Tecrübe yılı toplamı arttıkça çalışana ödenenen maaş miktarı artmaktadır, dolayısıyla tecrübe yılı toplamı değişkeninin hissedar açısından önem taşıyan bir değişken olduğu söylenebilir. Tecrübe yılı elenerek analiz tekrarlandığında küme sayısında küme yapısında kayda değer değişiklik gözlemlenmemiştir.

4.3 Mevcut Değer Akış Haritalama

Ele alınan projelerin tamamında aynı akış kullanılmaktadır. Yazılım geliştirme sürecindeki adımlar şu şekildedir.

1. Talep girişi: Yazılım geliştirme süreci son kullanıcının talep uygulaması üzerinden talebi girmesi ile başlar.
2. Maliyetlendirme: Yazılım geliştirme sürecinin ne kadar süreceğinin hesaplanmasıdır. Öncelikle analist kendi maliyetini belirler ve ardından yazılımcı kendi maliyetini belirler.
3. Analiz sürecinde son kullanıcının talebine uygun gereksinimler belirlenerek bir analiz dökümanı hazırlanır. Döküman son kullanıcının da onayından geçer ve eklemeler ve düzenlemeler sonucu, fonksiyonların da eklenmesi ile döküman yazılıma iletilir.
4. Yazılım geliştirme sürecinde kabaca tasarım, kodlama ve gözden geçirme adımları yer alır.
5. Test sürecinde hem fonksiyonel hem de kullanıcı test süreci yer almaktadır. Test süreci analistin projeyi test etmesi, test bulgularını yazılımcı ile paylaşması ardından bulguların düzeltilmesi ve en son tekrar test edilmesini içerir. Kullanıcı kabul testi kullanıcının uygulamayı test etmesi, elde ettiği bulguları paylaşması, bulguların düzetilmesi, analist tarafından test edilmesi ve son kullanıcı tarafından test edilmesi adımlarını içerir.
6. Canlı ortama geçiş süreci ile proje tamamlanmış olur.

Her bir kümede yer alan süreçlere ait adımlar için süreçlere ait veriler, proje bilgilerinin depolandığı Jira ortamından çekilerek ortalama işlem ve bekleme süreleri hesaplanmıştır. Küme-1 için oluşturulan mevcut değer akış diyagramı Şekil 2'de örnek olarak verilmiştir.



Şekil 2. Küme-1 için mevcut değer akış diyagramı

Tablo 6'da kümelerin ayrı ayrı işlem süreleri özetlenmiştir. Küme-1'de 175,2 günlük çevrim süresinin 27,5 günün işlem süresine ait olduğu görülmektedir. Küme 2'de 25,3 gün işlem süresine karşın 141,5 gün çevrim süresi bulunmaktadır. Küme-3 en yüksek işlem ve çevrim süresine sahiptir; 217,3 gün çevrim süresinin sadece 52,17 günü işlem süresine aittir. Küme-4 ise en düşük çevrim süresine sahip kümedir; 14,4 gün işlem süresi, 85,8 günlük çevrim süresi bulunmaktadır. Genel olarak tüm kümelerin işlem süresine karşılık çevrim süreleri, dolayısıyla bekleme süreleri yüksektir.

Her bir küme için mevcut değer akış haritaları oluşturulduktan sonra işlem sürelerinin ve bekleme sürelerinin uzun olmasına sebep olan faktörler (israflar) üzerinde durulmuştur.

Tablo 6
Süreçlere ait ortalama işlem süreleri (gün)

	Küme-1	Küme-2	Küme-3	Küme-4	*Ortalama*
Maliyetlendirme	0,97	0,51	0,52	0,52	*0,63*
SG maliyeti	0,05	0,16	0,04	0,04	*0,07*
YG maliyeti	0,52	0,10	0,18	0,18	*0,25*
Maliyet onaylama	0,24	0,12	0,18	0,18	*0,18*
Maliyet belirleme	0,16	0,13	0,12	0,12	*0,13*
Analiz	2,94	1,48	1,95	1,96	*2,08*
Analizde	1,96	0,98	1,58	1,48	*1,50*
Analiz onayı	0,01	0,04	0,15	0,11	*0,08*
Yönetici onayı	0,60	0,32	0,13	0,20	*0,31*
Kullanıcı onayı	0,26	0,10	0,05	0,13	*0,14*
Spek onay	0,11	0,04	0,04	0,04	*0,06*
Yazılım geliştirme	8,92	5,93	6,54	7,29	*7,17*
Tasarımda	7,80	4,28	5,15	6,1	*5,83*
Tasarım onay	0,04	0,10	0,01	0,01	*0,04*
Yazılım yapılıyor	0,98	0,69	1,1	0,34	*0,78*
Kod gözden geçirme	0,04	0,28	0,14	0,24	*0,18*
Yazılım tamamlandı	0,06	0,58	0,14	0,60	*0,35*
Test	5,62	3,55	1,51	4,68	*3,84*
Java-DB taşıması	1,89	0,32	0,50	0,91	*0,90*
Test	3,35	2,83	0,89	3,54	*2,65*
Kullanıcı test onayı	0,23	0,39	0,09	0,22	*0,23*
SG onay	0,15	0,01	0,03	0,01	*0,05*
Canlı ortama geçiş	0,26	0,04	0,05	0,02	*0,09*
Taşıma formu giriş	0,25	0,01	0,04	0,01	*0,08*
Taşındı	0,01	0,02	0,01	0,01	*0,01*

4.4 İsrafların belirlenmesi ve iyileştirme önerileri

Yazılım geliştirme projelerinde hatalar, ekstra süreçler, bekleme, ekstra özellikler, kısmen yapılmış iş, görev değişikliği ve hareketten kaynaklı olan israflar bulunmaktadır (Poppendieck ve Poppendieck 2003; Alahyari ve diğ., 2019). Uygulamada bu yedi israfla ilgili karşılaşılan durumlar ve israfları ortadan kaldırabilmek için iyileştirme önerileri aşağıda aktarılmıştır.

4.4.1 Hatalar

İncelenen firma süreçleri ele alındığında kullanıcı ihtiyaçlarının doğru belirlenmemesi sonucu analizde ortaya çıkan hatalar ve beraberinde yazılım sürecinde oluşan hatalar, yanlış kodlama ile ortaya çıkan fonksiyonel testler veya kullanıcı testlerinde ortaya çıkan hatalar, analist ve yazılımcının iyi iletişim içerisinde olmamasından kaynaklı hatalar ve sistem geliştirme talebi olmayan, teknik sorunlara yönelik olarak açılmış taleplerin uzunca süreler havuzda beklemesi göze çarpmaktadır.

Analiz sürecinde kullanıcıya "neden" sorusunun sorulması önemlidir, böylece asıl gereksinim açığa

çıkmış olacaktır. Bu doğrultuda yazılımcının da yanlış ihtiyaca yönelik kodlama yapmasının ve yanlış gereksinim test edilmesi gibi hataların önüne geçilmiş olacaktır.

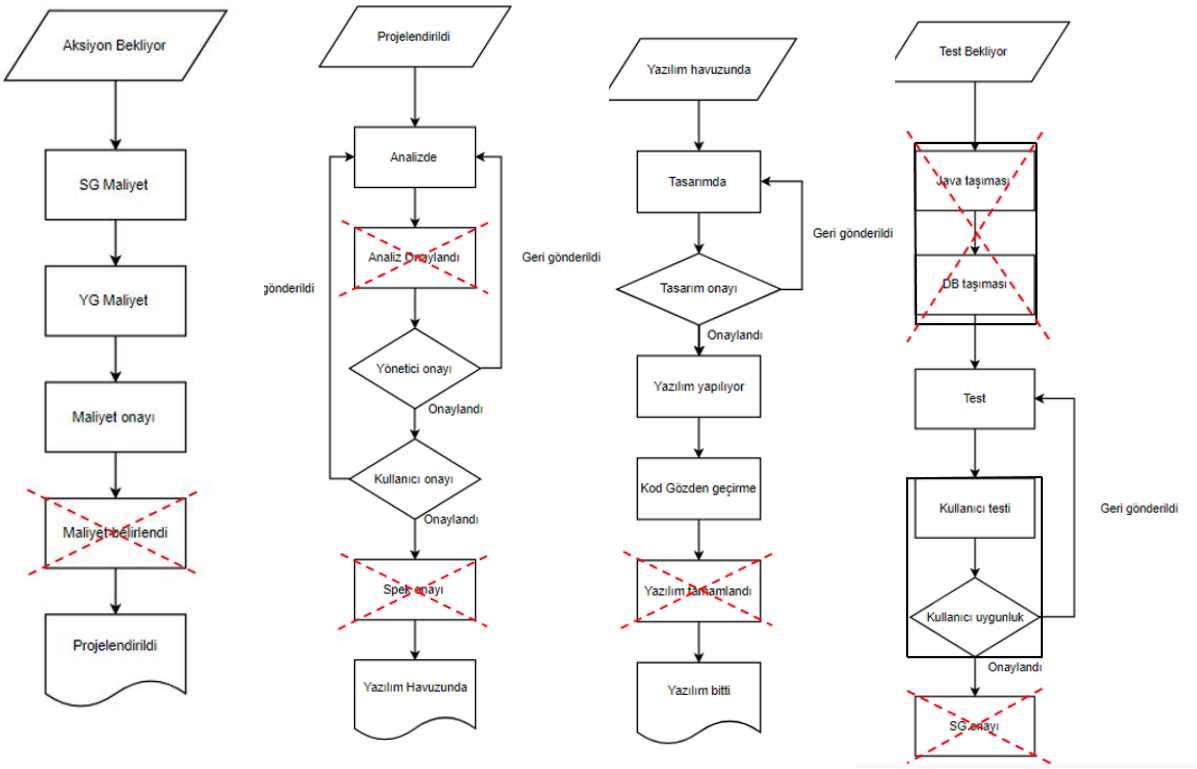
Yanlış açılmış taleplerin havuzda beklemesinin önleyebilmek için öncelikle birim yöneticilerinin talepleri detaylı inceleyerek uygunluk vermeleri gerekmektedir. Bunun yanında istenen projenin paydaşlarının daha önce böyle bir talepte bulunup bulunmadığı göz önüne alınmalıdır. Ayrıca talepler incelendiğinde bir kısmının teknik sorunları içerdiği, aslında sistem geliştirme talebi içerisinde yer almaması gerektiği gözlemlenmiştir. Bu tür durumların elenmesi halinde bekleme süresinin yarı yarıya azalacağını öngörebiliriz.

Yazılımcı ve analistlerin iletişim eksikliğinden veya yanlış aktarımdan kaynaklı hataların önüne geçebilmek için özellikle geniş kapsamlı projelerde analist ve yazılımcılar bir araya gelerek doküman üzerinden birlikte geçmelidirler.

Firmanın akışları incelendiğinde gereksiz süreç adımları görülmektedir. Örneğin analiz süreci detaylı incelenince analizin öncelikle analistin, sonra yöneticinin, arkasından kullanıcının ve en son tekrar yöneticinin onayından geçmesi gerekmektedir. Bu süreçler içerisinde analizi hazırlayan analistin tekrar onay vermesi ve yöneticinin en son tekrar onay vermesi süreçte israfa sebep olmaktadır. Bu iki adım elenirse Küme-1 için ortalama 0,12 gün, Küme-2 için 0,08 gün, Küme-3 için 0,19 gün ve Küme-4 için 0,15 gün kazanç sağlanacaktır.

Analiz ve yazılım süreci içerisinde dökümanların tamamlanmadan sürecin ilerletilememesi yerine süreç ilerletildikten sonra da döküman eklenebilmesi seçeneği sunulabilir. Tüm bunlar gözönünde bulundurularak analize başlama süresi ve analizde kalma süreleri kısaltılabilir. Şekil 3'te elenebilecek adımlar gösterilmiş ve gelecek durum akış diyagramı oluşturulurken bu yeni akışlar üzerinden hesaplama yapılmıştır.

4.4.2 Ekstra süreçler



Şekil 3. Gereksiz işlemlerin elenmesi

4.4.3 Bekleme

Yazılım geliştirme sürecinde çalışanların önceki işi beklemesi, onay bekleme, tekrarlar ve gözden geçirmeler diğer israflar olarak karşımıza çıkmaktadır.

Süreçte analistler hem analiz hem de test sürecinde yer almaktadır. Ayrı bir test ekibinin kurulması ile analistlerin analize ve teste başlamak için bekledikleri süre yarı yarıya inecektir. Birden fazla projenin aynı anda yürütüldüğü göz önünde bulundurulduğunda yazılım ve test süreçlerinin süreleri de yarı yarıya azaltılabilecektir.

Test sürecinin en başında analistin manuel yapması gereken test ortamına taşıma işlemleri yer almaktadır, bu işlemler otomatize edilerek beklemeden kaynaklı israf elenebilir.

Kullanıcı testi için beklemelerin en büyük nedeni bu işe özel personel ayrılmıyor olmasıdır. Kullanıcı testleri için o anda boş olan personelin rasgele kullanılması yerine bu iş için önceden plan yapılarak uygun personelin belirlenmesi beklemeleri azaltacaktır. Testlere başlamadan önce BT personeli tarafından kullanılacak test verilerinin net bir şekilde belirlenmesi tekrar gözden geçirmelerden kaynaklı beklemeleri ortadan kaldıracaktır.

4.4.4 Ekstra özellikler

Yazılımcıların kendilerine iletilen analiz dökümanına uymayarak ekstra özellikler eklemeleri da süreçte bir israfa sebep olmaktadır. Analiz dökümanında istenen özellikler dikkate alınarak kod yazılması bu israfın önüne geçecektir.

4.4.5 Kısmen yapılmış iş ve görev değişikliği

Aynı çalışanların eş zamanlı yürüyen birçok projede görev alıyor olması ve değişen proje öncelikleri kısmen yapılmış işlerin temel kaynağıdır. Analist ve yazılımcı elindeki işi kısmen tamamlamışken önceliklendirilen başka bir işe geçiş yaptığında bilişsel yük gereksiz şekilde artmaktadır. Bunun yanında çalışanlar projelerini yarım bırakarak başka bir projeye geçiş yaptıklarında bırakılan işi üstlenen çalışanın projeye alışma süresi de görev değişikliği ile gelen israf türü olarak gözlemlenmektedir. Önceliklendirmenin doğru zaman aralıkları ile yapılması bu israfın önüne geçebilir.

4.4.6 Hareket

Gün içinde yürütülen projelerin yanında canlı ortamda meydana gelen problemlerden kaynaklı olarak analist ve yazılımcılar harekete geçmek zorundadır. Bu da üzerinde çalıştıkları işi bırakarak sorunlarla ilgilenmeleri anlamına gelir. Proje sürelerine sorun çözümü ile ilgili süre öngörülerinin de eklenmesi düşünülebilir. Canlı ortamda meydana gelen problemler ile ilgilenmek üzere bir ekip kurulması, mevcutta yürütülen projelere yansıyan hareket israfını ortadan kaldırabilir.

Genel olarak özetleyecek olursak, israfı gereksiz kodlar, iyi tanımlanmamış ihtiyaçlar, bürokrasi, yavaş iç iletişim oluşturmaktadır. Müşteri ile sık sık iletişim, prototip oluşturma, kaynak sorununu anlama, ihtiyaçlar ve durum netleşinceye kadar bağlayıcı kararlar almama hataların önüne geçer. Takımların doğru liderler ile yönetilmesi, sistemin merkezi kavramlarının düzgün bir bütün olarak birlikte çalışmaları ve büyük işlerin küçük parçalara ayrılarak yönetilmesi gibi yalın düşünme yaklaşımları sürecin çevrim süresini azaltmada önemli faktörlerdir.

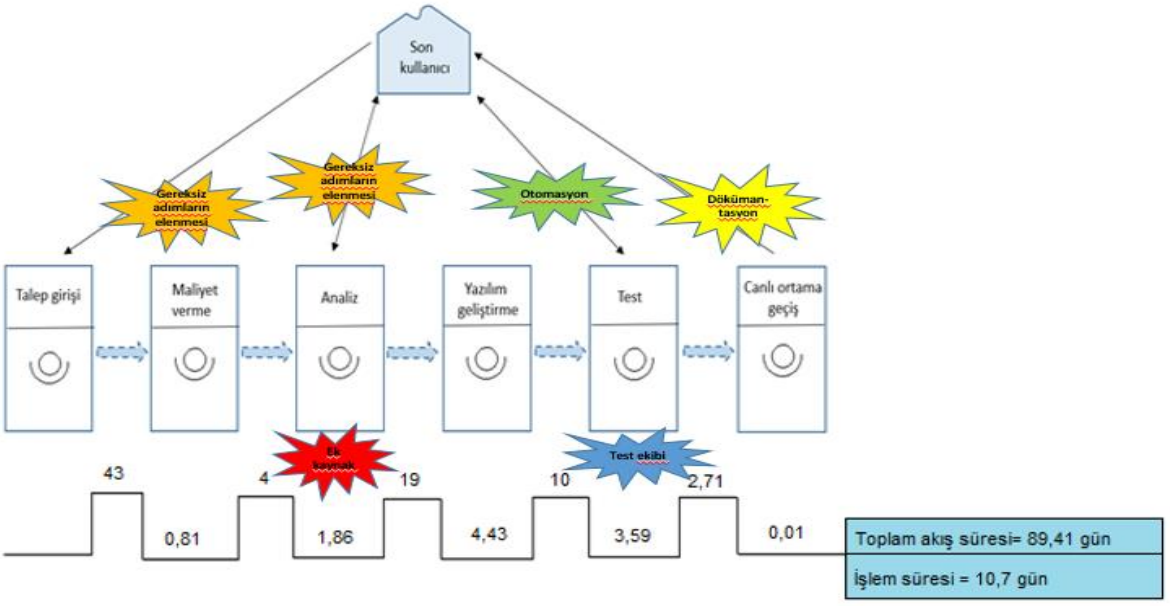
4.5 Gelecek Durum Değer Akış Haritalama

Değer akış haritaları ve süreç analizlerinin ardından aşağıda listelenen iyileştirmeler önerilmiştir.

- Kullanıcı tarafından girilen proje taleplerinin bir süzgeçten geçerek havuza yönlendirilmesi ile havuzda bekleme süresi yarı yarıya azaltılabilecektir.
- Maliyet onaylandıktan sonra gelen maliyet belirlendi adımı tamamen gereksiz ekstra bir süreç olduğundan süreçten çıkarılarak maliyetlendirme süreci kısaltılabilir. (Küme için 0,17 gün)
- Süreçler içerisinde analizi hazırlayan analistin tekrar onay vermesi ve yöneticinin en son tekrar onay vermesi işlemlerinin süreçten çıkarılması ile analiz adımı kısalacaktır. (Küme-1 için 0,12 gün)
- Kaynak sayısının 2 katına çıkarılması ile yazılıma başlama süresi ve yazılım geliştirme süreci yarı yarıya azaltılabilir.

- BT personeli tarafından kullanıcı test datalarının önceden belirlenmesi kullanıcı test için bekleme süresini azaltacaktır. Ayrıca test sürecinin en başında manuel yapılan test ortamına taşıma işlemleri otomatize edilerek bekleme süresi düşürülebilecektir. (Küme-1 için 1,89 gün)
- Test ekibi kurularak analistlerin test sürecindeki görevi son bulacak, böylece projeler test için beklemeyecektir. Kullanıcı tarafında ise teste ayrılacak bilgili personelin yer alması kullanıcı testinin süresini dolayısıyla toplam test süresini azaltacaktır.

En uzun bekleme süresine sahip olan Küme-1 için önerilen iyileştirmeler doğrultusunda Şekil 4'te görülen gelecek durum değer haritası çizilmiştir. İyileştirmeler sonucunda Küme-1 için toplam akış süresi ortalama 170,41 günden 89,41 güne kısalmış; işlem süresi ortalama 18,44 günden 10,7 güne inmiş ve işlemler için gereken kaynak ihtiyacı da azalmıştır.



Şekil 4. Küme-1 için gelecek durum değer akış haritası

5. Sonuçlar

Yazılım geliştirme projeleri için değer odaklı iyileştirme çalışmaları ile ilgili alanyazın oldukça kısıtlıdır. Bu çalışmalarda ortak olarak yazılım geliştirme alanında yalın yaklaşım ve değer akış haritalama yöntemi kullanılarak israflar belirlenmiş ve israfların ortadan kaldırılabilmesi için analizler ile süreç iyileştirmeleri yapılmıştır. Bu çalışmada da benzer şekilde yazılım geliştirme süreci için değer akış haritalama metodu kullanılmış ve israflar belirlenerek süreçte çevrim süresini kısaltacak iyileştirme önerileri sunulmuştur. Alanyazından farklı olarak bu çalışmada yazılım geliştirme projesi ürün aileleri, değer tanımından yola çıkarak belirlenen özellikler bazında iki adımlı kümeleme analizi ile gruplanarak belirlenmiştir.

Bu çalışmada çoklu proje ortamında yalın ilkeler ışığında değer odaklı iyileştirme için bir yöntem önerilmiştir. Bu amaçla yazılım geliştirme projeleri için tüm paydaşları dikkate alan değer tanımları yapılmış ve bu tanımlar çerçevesinde belirlenen kriterler ele alınan projeleri gruplamak için kullanılmıştır. Yazılım geliştirme projeleri için değer, doğruluk, güvenilirlik, veri bütünlüğü ve güvenliği, teslim zamanı, minimum maliyet, kullanım kolaylığı ve geliştirmeye açık olması şeklinde tanımlanmaktadır. Çoklu proje ortamında yalın ilkelerde önerildiği gibi ürünleri (yazılım geliştirme projesi) ürün ailesi bazında ele alabilmek için, söz konusu projeler değer tanımından yola çıkarak amaç, öncelik, maliyet gibi özellikleri bazında gruplanmıştır. Proje grupları iki adımlı kümeleme analizi ile belirlenmiştir. Analiz sonucu

elde edilen kümeler bir ürün grubu gibi ele alınmış ve değer akış haritalama gruplar bazında yapılmış, ortalama işlem ve bekleme süreleri belirlenmiştir. Süreçteki israflar ortaya konarak iyileştirme önerilerinde bulunulmuş ve gelecek durum haritaları oluşturularak yazılım geliştirme projeleri için değer odaklı iyileştirme alanına katkı sağlanmıştır.

Genel olarak incelendiğinde israfı iyi tanımlanmamış ihtiyaçlar, gereksiz kodlar, bürokrasi, yavaş iç iletişim oluşturmaktadır. Müşteri ile sık iletişim, görünürlüğün artması, prototip oluşturma, kaynak sorununu anlama, ihtiyaçlar ve kapasite netleşinceye kadar bağlayıcı kararlar almama israfların azaltılmasında etkendir. Takımların doğru liderler ile yönetilmesi, sistemin bir bütün olarak uyum içinde çalışması ve büyük işlerin küçük parçalara ayrılarak yönetilmesi gibi yalın düşünme yaklaşımları sürecin çevrim süresini azaltmada önemli faktörlerdir.

Yazılım geliştirme projelerinde değer odaklı iyileştirme çabaları işletme için maliyet ve verimlilik konusunda iyileşme sağlarken müşteri odaklılığı da beraberinde getirir. Bu yaklaşımla zamanında, daha iyi müşteri hizmeti ile daha kaliteli ürünler ortaya konabilir.

Yazılım geliştirme projelerinde çevrim süresini azaltmanın bir diğer ve daha maliyetli yolu ise kullanılan yazılım geliştirme metodunu değiştirmektir. İncelenen vakada şelale metodu kullanılmaktadır, bu da beklemeleri ve maliyeti artırmaktadır. CHAOS (www.standishgroup.com) raporuna göre zaman, bütçe ve müşteri memnuniyeti hedeflerini sağlama açısından daha başarılı olan çevik yaklaşıma geçilmesi önerilebilir. Ancak yazılım geliştirme metodunu değiştirmek özellikle kurumsal yapıda firmalar için oldukça maliyetli ve zaman alıcı bir geçiştir. Çalışanların uyum süreci ve değişime olan bakışları da geçişi zorlaştıran etkenlerden biridir. Bu çalışmada önerildiği gibi mevcut yapıyı değer odaklı iyileştirme çabası daha az maliyet ve dirençle daha yüksek fayda sağlayabilir.

Gelecek çalışmalarda yazılım geliştirme sürecinin değer boyutları müşteriler ve yazılım sağlayıcının yanısıra varsa diğer paydaşları da (örn. teknik destek hizmetleri) dikkate alacak şekilde daha kapsamlı olarak irdelenebilir. Yazılım geliştirme sürecindeki israflar ve bunları tetikleyen davranış ve alışkanlar tanımlanarak aralarındaki ilişkiler yapısal olarak modellenebilir. İsraflar ve/veya değerler arasındaki sebep sonuç ilişkileri örneğin Kısıtlar teorisi düsünce süreçleri ile irdelenebilir. Farklı yazılım geliştirme metodolojierinin israflar açısından farklılıkları araştırılabilir.

Araştırmacıların Katkısı

Bu araştırma Şeyda SERDARASAN'ın danışmanlığında Ebru ERTEK tarafından hazırlanan yüksek lisans tezinden üretilmiştir.

Çıkar Çatışması

Yazarlar tarafından herhangi bir çıkar çatışması beyan edilmemiştir.

Kaynaklar

Ahmad, A., ve Khan, S. S. (2019) Survey of State-of-the-Art Mixed Data Clustering Algorithms. IEEE Access, 7, 31883-31902. doi: https://doi.org/10.1109/ACCESS.2019.2903568

Aktaş, E. U., Altunel, H. Elalmış, H. I. Nişancı, S. ve Yılmaz, C. (2018) *Endüstriyel bağlamda yazılım olay kaydı yönetim sürecinin iyileştirilmesi*. In: 12. Ulusal Yazılım Mühendisliği Sempozyumu , Istanbul, Turkey.

Alahyari, H., Gorschek, T., ve Svensson, R. B. (2019). An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations. *Information and Software Technology,* 105, 78-94. Doi: https://doi.org/10.1016/j.infsof.2018.08.006

Alahyari, H., Svensson, R. B., ve Gorschek, T. (2017). A study of value in agile software development organizations. *Journal of Systems and Software*, 125, 271-288. Doi: https://doi.org/10.1016/j.jss.2016.12.007.

Ali N, Petersen, K. ve Schneider, K. (2015). Evaluation of simulation-assisted value stream mapping for software product development: Two industrial cases*, Information and Software Technology*, 68, 45-61. Doi: https://doi.org/10.1016/j.infsof.2015.08.005

Ali N, Petersen, K. ve Schneider, K. (2016). FLOW-assisted value stream mapping in the early phases of large-scale software development, *The Journal of Systems and Software*, 111, 213-227. Doi: https://doi.org/10.1016/j.jss.2015.10.013

Anand, G., Chandrashekar, A. ve Narayanamurthy, G. (2014). Business process reengineering through lean thinking: A case study, *Journal of Enterprise Transformation,* 4 (2), 123-150. Doi: https://doi.org/10.1080/19488289.2013.879681

Aurum, A., ve Wohlin, C. (2007). *A value-based approach in requirements engineering: explaining some of the fundamental concepts*. In International Working Conference on Requirements Engineering: Foundation for Software Quality (pp. 109-115). Springer, Berlin, Heidelberg. Doi: https://doi.org/10.1007/978-3-540-73031-6_8

Bauch, C. (2004). *Lean product development: making waste transparent*, (Doktora Tezi), MIT Sociotechnical Systems Research Center (SSRC), Erişim Adresi: http://hdl.handle.net/1721.1/81429

Boehm, B. (2006). Some future trends and implications for systems and software engineering processes. *Systems Engineering*, 9(1), 1-19. Doi: https://doi.org/10.1002/sys.20044

Boehm, B., (1988) A Spiral Model for Software Development and Enhancement, *Computer*, vol. 21, no. 5, May 1988, pp. 61–72. Doi: https://doi.org/10.1109/2.59

Cooke, J. L. (2016). Agile: an executive guide: real results from IT budgets. IT Governance Ltd.

Dingsoyr, T., Nerur, S., Balijepally, V., ve Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software*, 85 (6), 1213-1221. Doi: https://doi.org/10.1016/j.jss.2012.02.033

Dyba, T., ve Dingsoyr, T. (2009). What do we know about agile software development?. *IEEE software*, 26(5), 6-9. Doi: https://doi.org/10.1109/MS.2009.145

Ebert, C. ve Dumke, R. (2007). Software measurement: Establish - Extract - Evaluate - Execute, *Springer-Verlag Berlin Heidelberg*. Doi: https://doi.org/10.1007/978-3-540-71649-5

Elibol, M , Selçukcan Erol, Ç . (2017). Scrum Metodu Kullanılarak Bir Mobil Uygulama Geliştirme Sürecinin Gerçekleştirilmesi . *Bilişim Teknolojileri Dergisi* , 10 (2) , 169-176 . Doi: https://doi.org/10.17671/gazibtd.309299

Futcher, L. ve von Solms, R. (2008). Guidelines for secure software development. In Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology (SAICSIT '08). *Association for Computing Machinery*, New York, NY, USA, 56–65. Doi: https://doi.org/10.1145/1456659.1456667

Haksever, C. ve Chaganti, R. (2004). A Model of Value Creation: Strategic View, *Journal of Business Ethics,* 49, 291-305. Doi: https://doi.org/10.1023/B:BUSI.0000017968.21563.05

Janzen, D., ve Saiedian, H. (2005). Test-driven development concepts, taxonomy, and future direction. *Computer*, 38(9), 43-50. Doi: https://doi.org/10.1109/MC.2005.314

Khan, P. M. ve Beg, M. M. S. S. (2013), *Extended Decision Support Matrix for Selection of SDLC-Models on Traditional and Agile Software Development Projects*, 2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT), Rohtak, pp. 8-15. Doi: https://doi.org/10.1109/ACCT.2013.12

Levina, N., ve Ross, J. W. (2003). From the vendor's perspective: exploring the value proposition in information technology outsourcing. *MIS Quarterly*, 331-364. Doi: https://doi.org/10.2307/30036537

Mele C., ve Polese F. (2011) Key Dimensions of Service Systems in Value-Creating Networks. In: Demirkan H., Spohrer J., Krishna V. (eds) The Science of Service Systems. Service Science: *Research and Innovations in the Service Economy*. Springer, Boston, MA. Doi: https://doi.org/10.1007/978-1-4419-8270-4_3

Mujtaba, S., Feldt, R., Petersen, K., (2010), *Waste and lead time reduction in a software product customization process with value stream maps*, Proceedings of the Australian Software Engineering Conference, ASWEC, art. no.

5475062, pp. 139-148. Doi: https://doi.org/10.1109/ASWEC.2010.37

Murphy, G.C. ve Kersten, M. (2020). Towards Bridging the Value Gap in DevOps. *Lecture Notes in Computer Science*, 12055, 181-190. Doi: https://doi.org/10.1007/978-3-030-39306-9_13

Norusis, M. J. (2012). IBM SPSS statistics 19 advanced statistical procedures companion (p. 444*). Upper Saddle River*, NJ: Prentice Hall.

Ohno, T. (1988). Toyota production system: beyond large-scale production. CRC Press.

Özcan-Top, Ö. ve McCaffery, F. (2018). A hybrid assessment approach for medical device software development companies, *Journal of Software: Evolution and Process,* 30 (7), e.1219. Doi: https://doi.org/10.1002/smr.1929

Papadopoulos, G. (2015). Moving from traditional to agile software development methodologies also on large, distributed projects, *Procedia - Social and Behavioral Sciences*,175, 455-463. Doi: https://doi.org/10.1016/j.sbspro.2015.01.1223

Poppendieck, M. (2011). Principles of lean thinking. IT Management Select, 18(2011), 1-7.

Poppendieck, M., ve Poppendieck, T. (2003). Lean Software Development: An Agile Toolkit. Addison-Wesley.

Pressman R., ve Maxim, B. (2020) Software Engineering: A Practitioner's Approach, 9th Ed., McGraw Hill.

Racheva, Z., Daneva, M., ve Sikkel, K. (2009*). Value creation by agile projects: Methodology or mystery?.* in International Conference on Product-Focused Software Process Improvement (pp. 141-155). Springer, Berlin, Heidelberg.

Rother, M., ve Shook, J. (2003). Learning to see: value stream mapping to add value and eliminate muda. Lean Enterprise Institute.

Sandström, S., Edvardsson, B., Kristensson, P. ve Magnusson, P. (2008), Value in use through service experience, Managing Service Quality*: An International Journal,* Vol. 18 No. 2, pp. 112-126. Doi: https://doi.org/10.1108/09604520810859184

Sarstedt, M., ve Mooi, E. (2019) A Concise Guide to Market Research: The Process, Data, and Methods Using IBM SPSS Statistics. 3rd ed. *Springer.*

Sedano, T., Ralph, P., ve Péraire, C. (2017). *Software development waste.* In 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), pp. 130-140. Doi: https://doi.org/10.1109/ICSE.2017.20

Staron, M., Meding, W. ve Caiman, M. (2013). Improving completeness of measurement systems for monitoring software development workflows. *Lecture Notes in Business Information Processing*, 133, 230-243. Doi: https://doi.org/10.1007/978-3-642-35702-2_14

Stephens, R. (2015). Beginning software engineering. John Wiley ve Sons.

Vargo, S. L., ve Lusch. R. F. (2004). Evolving to a new dominant logic for marketing. *Journal of Marketing,* No. 68 (1):1-17. Doi: https://doi.org/10.1509/jmkg.68.1.1.24036

Vargo, S. L., Maglio, P.P., ve Akaka M. (2008). "On value and value co-creation: A service systems and service logic perspective." *European Management Journal,* No. 26 (3):145-152. Doi: https://doi.org/10.1016/j.emj.2008.04.003

Vijayasarathy, L., ve Butler, C. (2016). Choice of Software Development Methodologies. IEEE Computer Society - *IEEE Software*, September/October, 86-94. Doi: https://doi.ieeecomputersociety.org/10.1109/MS.2015.26

Womack, J.P. and Jones, D.T., (1996). Lean Thinking: Banish Waste and Create Wealth in Your Corporation, Simon ve Schuster, New York.

Yadav, R., Mittal, M. L., ve Jain, R. (2018). Adoption of lean principles in software development Projects. *International Journal of Lean Six Sigma*. Doi: http://dx.doi.org/10.1108/IJLSS-03-2018-0031