İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM PROJE GELİŞTİRME SÜRECİNDE DEĞER ODAKLI İYİLEŞTİRME VE BİR UYGULAMA

YÜKSEK LİSANS TEZİ

Ebru ERTEK

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Programı

MAYIS 2020

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM PROJE GELİŞTİRME SÜRECİNDE DEĞER ODAKLI İYİLEŞTİRME VE BİR UYGULAMA

YÜKSEK LİSANS TEZİ

Ebru ERTEK
(507161131)

Endüstri Mühendisliği Anabilim Dalı

Endüstri Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Şeyda SERDAR ASAN

MAYIS 2020

İTÜ, Fen Bilimleri Enstitüsü'nün 50716131 numaralı Yüksek Lisans Öğrencisi Ebru ERTEK, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "YAZILIM PROJE GELİŞTİRME SÜRECİNDE DEĞER ODAKLI İYİLEŞTİRME VE BİR UYGULAMA" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı : **Dr. Öğr. Üyesi Şeyda SERDAR ASAN** .....

Jüri Üyeleri : **Prof. Dr. Şule Itır ŞATOĞLU** .....
İstanbul Teknik Üniversitesi

Prof. Dr. Ahmet BEŞKESE .....
Bahçeşehir Üniversitesi

Teslim Tarihi **: 06 MAYIS 2020**
Savunma Tarihi **: 22 MAYIS 2020**



Canım eşime,

ÖNSÖZ

Bu çalışmada Yazılım Geliştirme Süreçlerinde Değer Odaklı İyiletişme konusu incelenmiş ve Finans sektründe bir uygulama yapılmıştır. Tez çalışmam boyunca başta yardımlarını esirgemeyen Şeyda Serdar Asan hocama, beraberinde sevgili eşime ve aileme destekleri için teşekkürlerimi iletiyorum.

Mayıs 2020

Ebru Ertek
Endüstri Mühendisi

İÇİNDEKİLER

Sayfa

ÖNSÖZ ..... **vii**
İÇİNDEKİLER ..... **ix**
ÇİZELGE LİSTESİ ..... **xi**
ŞEKİL LİSTESİ ..... **xiii**
ÖZET ..... **xv**
SUMMARY ..... **xvii**
1. GİRİŞ **1**
2. LİTERATÜR ARAŞTIRMASI **5**
2.1. Değerin Tanımı 5
2.2. Değerin Yazılım Projeleri İçin Tanımı 8
2.3. Yazılım Geliştirme Süreçlerinde Değer Akış Haritalama 10
2.4. İsrafın Yazılım Projeleri İçin Tanımı 12
3. METODOLOJİ **15**
3.1. Metodolojinin Adımları 15
3.2. İki Adımlı Kümeleme Analizi 17
4. UYGULAMA **21**
4.1. Değer Tanımı ve Proje değerlendirme Kriterlerinin Belirlenmesi 21
4.2. Yazılım Projelerinin Gruplanması 22
4.3. Mevcut Değer Akış Haritalama 27
4.4. İsrafların belirlenmesi ve iyileştirme önerileri 34
4.4.1.Hatalar 34
4.4.2.Ekstra süreçler 35
4.5. Gelecek Durum Değer Akış Haritalama 39
5. SONUÇ **41**
ÖZGEÇMİŞ **49**

ÇİZELGE LİSTESİ

Sayfa

Çizelge 2.1: Paydaşlara göre değer tanımları **7**
Çizelge 4.1 : Sınıflandırmada kullanılan bağımsız değişkenler **22**
Çizelge 4.2 : Değişkenler arası korelasyon matrisi **23**
Çizelge 4.3 : İki adımlı kümeleme sürecinin sonuçları **23**
Çizelge 4.4 : Projelerin kümelere dağılımı **24**
Çizelge 4.5 : Projelerin özet profilleri **25**
Çizelge 4.6 : Süreçlere ait ortalama süreler **28**

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : Çevik ve klasik projelerin karşılaştırması **3**
Şekil 2.1 : Değerin yaratılması **6**
Şekil 3.1 : Uygulama adımları akış diyagramı **17**
Şekil 4.1 : Değişkenlerin her bir küme içerisinde önem sırası **25**
Şekil 4.2 : Kümeleme yapılırken değişkenlerin önem sıralamaları **26**
Şekil 4.3 : Maliyetlendirme ve Analiz süreçleri **29**
Şekil 4.4 : Yazılım geliştirme ve Test süreçleri **30**
Şekil 4.5 : Taşıma süreci **30**
Şekil 4.6 : 1. Küme için mevcut değer akış diyagramı **31**
Şekil 4.7 : 2. Küme için mevcut değer akış diyagramı **31**
Şekil 4.8 : 3. Küme için mevcut değer akış diyagramı **32**
Şekil 4.9 : 4. Küme için mevcut değer akış diyagramı **32**
Şekil 4.10 : Fazladan işlemlerin çıkarılması ile maliyet ve analiz süreci **36**
Şekil 4.11 : Fazladan işlemlerin çıkarılması ile maliyet ve analiz süreci **37**
Şekil 4.12 : Fazladan işlemlerin çıkarılması ile taşıma süreci **37**
Şekil 4.13 : Gelecek durum değer akış haritası **40**

YAZILIM PROJE GELİŞTİRME SÜRECİNDE DEĞER ODAKLI İYİLEŞTİRME VE BİR UYGULAMA

ÖZET

Yazılım geliştirme metodolojisi, bilgi sistemleri geliştirme sürecini düzenlemek ve kontrol etmek için kullanılır. Tek bir sistem geliştirme metodolojisi tüm projeler tarafından kullanılmak için uygun değildir. Mevcut metodolojilerin her biri, çeşitli teknik, organizasyonel, proje ve ekip konularına dayanarak, belirli proje türlerine uygun biçimde şekillenmiştir. Yazılım geliştirme süreci veya yaşam döngüsü, bir yazılım ürününün geliştirilmesine uygulanan bir yapıdır. Her biri süreç boyunca gerçekleşen çeşitli görevlere veya faaliyetlere yaklaşımları tanımlayan bu tür işlemler için birkaç model vardır.

Üretim alanında başarılı sonuçlar elde edilmesiyle birlikte yalın süreç iyileştirme yaklaşımlarının Yazılım Geliştirme alanında da uygulanmaya alınması ile yazılım geliştirme süreçlerinde yaşanan sorunların aşılmasına yönelik adımlar atılmaya başlanmıştır. Geçmişi 1960'lara dayanan Yalın yaklaşımları da üretim alanında büyük kazanımlar sağlanmasından dolayı yazılım geliştirme alanı da dahil olmak üzere birçok başka alan için kullanımı değerlendirilen yöntemlerden birisi olmuştur. Temelde atık ve israfın yok edilmesine dayalı olan Yalın yaklaşımları Değer Akış Haritalama yöntemi ile başlamaktadır. Yalın prensipler iyi tanımlanmış ve üretimde geniş bir uygulanabilirliğie sahip olsa da bilgi teknolojisine olan yayılımı hala sınırlıdır. Bu çalışma bir vaka çalışması yaklaşımına dayanarak bir firmanın BT alanında Değer Akış Haritalama yöntemini kullanarak iyileştirme girişimi uygulamasıdır.

Bu çalışmanın amacı, yazılım geliştirme projesinde mevcut değer akış haritasından yola çıkarak israfları belirleyebilmek ve bu israfları ortadan kaldırabilecek ve süreç iyileştirmeleri sağlayacak öneriler sunarak daha kısa çevrim süreli gelecek durum değer akış haritasını oluşturabilmektir. Bu amaçla yazılım ürünü ve yazılım geliştirme süreci değer odaklı olarak ele alınmış ve yazılım geliştirme sürecine iyileştirmeye yönelik olarak değer akışı haritalama yöntemi uygulanmıştır. Mevcut ve gelecek durum haritalarının oluşturulabilmesi için öncelikle yazılım geliştirme süreçleri için değer tanımı yapılmış ve bu değer tanımından yola çıkarak projelerin sınıflandırılabilmesi adına kriterler belirlenmiş ve projeler gruplandırılmıştır. Her bir grup için mevcut durum değer akış haritası çizilmiştir. Bu haritalardan yola çıkarak bekleme süresini uzatan faaliyetler yani israflar tespit edilerek çevrim süresini kısaltacak ve süreci iyileştirecek iyileştirme öneleri sunulmuş ve bu doğrultuda en uzun bekleme süresine ait proje kümesi için gelecek durum değer akış haritası çıkarılmıştır.

Çevrim süresini uzatacak israfları gereksiz kodlar, iyi tanımlanmamış ihtiyaçlar, bürokrasi, yavaş iç iletişim oluşturmaktadır. Müşteri ile sık sık iletişim, prototip oluşturma, kaynak sorununu anlama, gerçekler belirleninceye kadar bağlayıcı kararlar alınmaması 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.

Yazılım geliştirme projeleri için değer odaklı iyileştirme çalışmaları oldukça kısıtlıdır, bu çalışmada hem iki adımlı kümeleme analizi yapılmış hem de değer odaklı iyileştirme yapılarak mevcut ve gelecek durum haritalarının oluşturularak yazılım geliştirme projeleri için değer odaklı iyileştirme alanında kaynak sağlanmıştır. Bu araştırma, Bilgi Teknolojileri departmanı için yapılmıştır. Fakat şirket içerisinde insan kaynakları, finans, satış ve pazarlama, müşteri hizmetleri ve destek, vb. diğer alanlara da genişletilebilir. Maliyet ve benzeri performans ölçütlerinde önemli miktarda tasarrufla birlikte şirketin tamamındaki işleri yönetmede etkinlik sağlayabilir. Zamanında ve daha iyi müşteri hizmeti ile kaliteli ürünler ortaya çıkarılabilir.

VALUE ORIENTED IMPROVEMENT IN THE SOFTWARE PROJECT DEVELOPMENT AND AN APPLICATION

SUMMARY

Software development methodology is used to regulate and control the information systems development process. A single system development methodology is not suitable for use by all projects. Each of the available methodologies is tailored to specific project types, based on a variety of technical, organizational, project and team issues. A software development process or life cycle is a structure that applies to the development of a software product. There are several models for such processes, each of which defines approaches to the various tasks or activities that occur during the process.

With the successful results obtained in the field of production, the implementation of lean process improvement approaches in the field of Software Development has started to take steps to overcome the problems experienced in software development processes. Lean approaches, which date back to the 1960s, have been one of the methods used for many other fields, including software development, due to the huge gains in the field of production. Lean approaches, which are mainly based on waste and waste elimination, begin with Value Flow Mapping. Although lean principles are well defined and have broad applicability in production, their spread to information technology is still limited. This study is a case study approach based on a firm's attempt to improve the IT field using Value Flow Mapping.

The aim of this study is to present a solution proposal that provides a quality software product that will provide customer satisfaction with more effective communication and visibility to the end user, simultaneously reducing the total supply time and increasing resource utilization efficiency. For this purpose, the software product and software development process are handled with a focus on value, and a value stream mapping method has been applied to improve the software development process. In order to create current and future status maps, firstly, the definition of value has been made for software development processes and criteria have been determined and the projects have been grouped in order to classify the projects based on this value definition. The current state value flow map was drawn for each group. Based on these maps, activities that prolong the waiting time, namely waste, were identified and improvement measures were offered to shorten the cycle time and improve the process, and in this direction, a future value flow map was created for the project set for the longest waiting period.

Firstly, in the literature, definitions are made based on the value definitions related to software development and that take into account the current status and expectations of the software provider. It is important that the defined value is converted into measurable and / or observable dimensions and that the data of them are accessible. Value is defined separately in terms of shareholder, employee and customer. Shareholder: The company has financial shares and expects investments to gain value

or generate income. Employee: People working for a company rely on wages or salaries for their livelihood. In addition, health insurance, pensions, paid vacation time, family sick leave, childcare facilities, etc. Such social benefits constitute an important part of compensation packages.Customer: Customers expect certain benefits from the products they purchase. In general, they remain loyal to the company as long as their needs are met at a reasonable price. If customers cannot obtain the products they need from another company at an affordable price, they may be concerned with the company's survival and well-being.

Software value dimensions are listed in the literature as follows. Accuracy comes together with analysts and project stakeholders before developing a software, and a specification is prepared in line with the requirements. It is a definition of value that the prepared software is true to the created specification. This is to some extent a measure of whether the software is functioning properly. Reliability is how often an error will occur in the process after the software project is put into practice, that is, in a live environment. Data integrity and security, large programs or program systems often have a responsibility to ensure a permanent mass of data. Data may need to provide consistency limitations. If so, the software should ensure that it does not violate these restrictions. In addition, it may be the software's responsibility to limit access to data. It is especially important to limit the changes of the data. And changes with serious consequences should not be too easy. Delivery time, delivery on time - the buyer of the software always waits for the delivery of the product on time. When the buyer is a business firm, he needs the software on the date given to fulfill his commitments. In case the software cannot be delivered before the deadline, the buyer may stop receiving the product and turn to another company. Therefore, it may be preferable to deliver the software incorrectly instead of delivering it after the deadline. The minimum cost, the buyer will want to pay the lowest price to achieve the desired functionality and quality. Ease of use is that it offers a clear interface to the user and the availability of the application is easy. Being open to development should be able to meet the additional requests placed on the software.

In practice, the IBM SPSS Two-Step Cluster Analysis technique is used to cluster projects because we have both categorical and metric data to evaluate software projects. The variables of priority and purpose used in the analysis are categorical, cost, effort, resource, number of additional requests, number of updates and total years of experience are metric variables. Although the two-step analysis of the variables did not meet the normality and multinomality assumptions, the analysis continued because the method was robust.

After the projects are grouped, a value stream map is drawn. The value stream mapping method is the lean planning tool used to visualize the value stream and is considered the main approach to identify the waste involved in a process. Every step that creates a process is classified as an activity that adds or adds value.

On the current value stream map, activities that create and do not create value within the framework of value definitions are examined and waste is revealed. Improvement suggestions are provided for eliminating or reducing waste.

The future value stream map, which will be achieved with the implementation of the proposed improvement suggestions, has been drawn and new processing times have been calculated.

Unnecessary codes, undefined needs, bureaucracy, and slow internal communication are used to prolong cycle time. Frequent communication with the customer,

prototyping, understanding the source problem, and not making binding decisions until the facts are determined, prevent mistakes. Lean thinking approaches such as managing teams with the right leaders, working together as the central concepts of the system as a whole, and managing big jobs in small pieces are important factors in reducing the cycle time of the process.

Another and more costly way to reduce cycle time is changing the software development method used. In the case we examined, the waterfall method, also known as the classical approach, is used. The most important feature of this method is that one phase cannot be passed to another phase. In this case, either the employee is waiting in the intermediate passes or the project is waiting because the employee has a workload. Also, returning from one phase to another requires high cost.

As an alternative method, Agile software development method can be suggested. Agile method focuses on producing valuable output in a short time. Projects are divided into small pieces and more than one process is carried out in parallel. This situation plays a big role in detecting errors in a short time and resolving them. Moreover, its biggest advantage over the classical method is that it is quite flexible for the changing requirement. Due to the constant feedback to the customer, it is greatly reduced in misunderstandings. However, changing the method of software development is particularly difficult in companies with a corporate structure. It is costly because it requires new technologies and training and the adaptation process of the employees is one of the factors that make the transition difficult.

Value-oriented improvements are limited for software development projects. In this study, a two-step cluster analysis was performed and a resource was provided in the field of value-oriented improvement for software development projects by creating value-oriented improvements by creating current and future status maps. This research was carried out for the Information Technologies department. But human resources, finance, sales and marketing, customer service and support, etc. can be extended to other areas. With significant savings in cost and similar performance criteria, it will provide efficiency in managing the entire company. Quality products can be created on time and with better customer service.

1. GİRİŞ

Yazılım geliştirme süreci veya yaşam döngüsü, bir yazılım ürününün geliştirilmesine uygulanan bir yapıdır. Yazılım geliştirme süreç metodolojisi, yazılım ve bilgi sistemleri geliştirme sürecini planlamak, yürütmek ve kontrol etmek için kullanılan bir çerçevedir. Zaman içinde her birinin kendine özgü güçlü ve zayıf yönleri olan çok çeşitli metodoloji ortaya konmuştur ve bunlar gittikçe daha fazla sayıda işletme tarafından yazılım geliştirme süreçlerinde uygulanmaktadır. Stephens (2015) yazılım süreç modellerini kestirimci (predictive) modeller (örn. Şelale, V-Modeli), yinelemeli (iterative) modeller (örn. Spiral, Cleanroom) ve çevik (agile) modeller (örn. Scrum, XP, Lean) olmak üzere üç başlık altında sınıflandırmaktadır. Süreç geliştirme metodolojilerindeki çeşitliliğin gerekliliği, tek bir süreç geliştirme metodolojisinin her türlü yazılım geliştirme projesinde ve organizasyonunda kullanıma uygun olmamasından 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 (Waterfall), çevik tümleşik süreç (Agile unified process), Scrum, Test odaklı geliştirme (Test-driven development), Yalın (Lean software development) gibi yöntemler sıralanmaktadır (Vijayasarathy ve Butler, 2016).

Süreç modellerinin yanısıra yazılım sürecini iyileştirmek değerlendirmek için kullanılan olgunluk modelleri de bulunmaktadır; CMMI (CMMI, 2010) ve ISO/IEC 33001:2015 (SPICE) uluslararası düzeyde belgelendirilen ve yaygın olarak kullanılan modellerdir. Bağımsız değerlendirmeler, kuruluşları süreçlerini nasıl tanımladıkları ve uyguladıklarına göre yazılımları ne kadar iyi oluşturdukları konusunda derecelendirmek için kullanılmaktadır.

Bir yazılım geliştirme projesinde (yazılım yaşam döngüsü) ihtiyaçların belirlenmesi, tasarım, test, uygulamaya alma (taşıma/teslimat) ve bakım adımlarının gerçekleşmesi gerekmektedir. Süreç modellerini birbirinde ayıran temel fark bu adımların nasıl ve

hangi sırayla ele alındığı konusundadır. Sık kullanılan metodolojilerin işleyişi ve avantajları kısaca aşağıdaki gibi özetlenebilir.

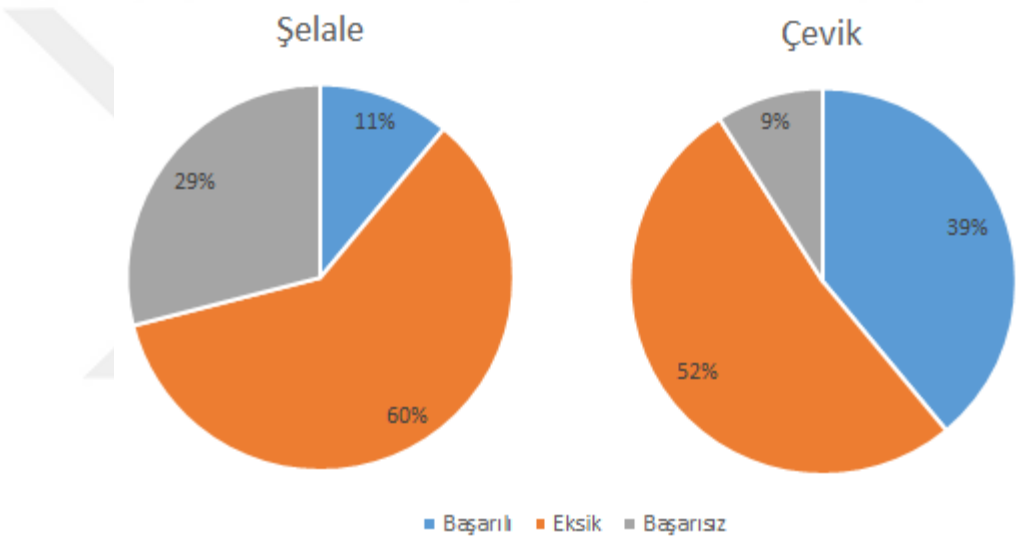
Şelale yöntemi, diğer deyişle klasik yazılım geliştirme yaklaşımı, temelde bir süreç adımı bitmeden diğer adıma başlanmamasıdır. Bir adım tamamlandıktan sonra tekrar o adıma geri dönüş nadir gerçekleşen bir durumdur. Bu yüzden de tüm adımların olabildiğince plana uygun işlemesi gerekir. İhtiyaçların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

Spiral model, Boehm (1988) tarafından önerilmiş, prototiplemenin yinelemeli yapısını şelale modelinin kontrollü ve sistematik yönleriyle birleştiren iteratif bir yazılım süreç modelidir. 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 şelale modelinden daha etkindir. Büyük ölçekli projelerde daha kullanımı önerilir (Stephens, 2015).

Çevik yaklaşımlardan biri günümüzde giderek popüler hale gelen Scrum yaklaşımdır. Scrum (ismini bir ragbi maçı sırasında gerçekleşen bir pozisyondan 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. Her adım içinde iş adımları, sprint adı verilen bir süreç deseni içinde gerçekleşir. Her bir süreç adımı için gerekli olan sprint sayısı ürün karmaşıklığına ve büyüklüğüne bağlı olarak değişiklik gösterir. Bir sprint içinde yürütülen faaliyet eldeki probleme uyarlanır ve Scrum ekibi tarafından gerçek zamanlı olarak tanımlanır ve iteratif olarak yürütülür ve her sprint sonunda uygulama canlıya alınabilir. Bir özellik geliştirilirken herhangi bir sorun meydana gelirse bu sorunun diğer kısımları geliştirmeye engel oluşturma riski azaltılmış olur, böylece risk dağıtılır (Elibol ve Erol, 2017). Scrum zaman kısıtı, ihtiyaçlardaki değişkenlik, ve kritiklik gibi koşullarda iyi sonuçlar vermektedir (Pressman ve Maxim, 2020).

Bir diğer çevik yöntem ise yalın yazılım geliştirme yaklaşımıdır. Yazılım geliştirmede yalın yaklaşım, yalın düşünme prensiplerinin yazılım mühendisliğine uygulanmasını temel alır (Poppendieck ve Poppendieck, 2003). Bu yaklaşım da özelliklerin küçük parçalara bölünmesi ve geliştirmenin iteratif olarak yürütülmesi konusunda Scrum

modeline benzerlik gösterir. Yalın yöntem, değer yaratan gereksinimleri tanımlamaya ve her bir faaliyete yalnızca gerekli (katma değer yaratan) bileşenlerin girmesini sağlamaya odaklanır. Yalın yöntemin temel yaklaşımı israfı engellemektir, bu sebeple gerekli olmayan hiçbir özellik geliştirilmez. Şekil 1.1'de çevik ve klasik projelerin zaman, bütçe ve müşteri memnuniyeti hedeflerini sağlama başarıları görülmektedir (CHAOS, 2015). Tüm hedefleri planlandığı şekilde karşılayan projeler başarılı olarak nitelendirilirken, proje tamamlanmasına rağmen müşteri memnuniyeti tam sağlanamamışsa ilgili proje eksik olarak nitelendirilmiştir. CHAOS (2015) raporu verilerine bakarak çevik yaklaşımın başarı oranlarının genel anlamda daha yüksek olduğu yorumu yapılabilir.



Şekil 1.1 : Çevik ve klasik projelerin karşılaştırması, CHAOS (2015)'ten uyarlanmıştır.

Yazılım geliştirme sürecinde kullanılacak süreç yaklaşımını seçmek, 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 ve eş zamanlı olarak toplam tedarik süresini azaltan ve kaynak kullanılım verimliliğini artıran bir çözüm önerisi ortaya koymaktır. Bu amaçla yazılım ürünü ve yazılım geliştirme süreci değer odaklı olarak ele alınmış ve yazılım geliştirme sürecine iyileştirmeye yönelik olarak değer akışı haritalama yöntemi uygulanmıştır.

Bu kapsamda öncelikle yazılım geliştirme süreçleri için değer tanımı yapılmış ve bu değer tanımından yola çıkarak projelerin sınıflandırılabilmesi adına değer odaklı kriterler belirlenmiştir. Sonrasında İki adımlı kümeleme (TwoStep Cluster) analizi uygulanarak kümeler oluşturulmuştur. Yazılım sürecinde değer odaklılığı desteklemek ve uygulamak için tercih edilen bir yol olarak karşımıza Değer Akış Haritalama metodu çıkmaktadır. Değer Akış Haritalama israf ve iyileştirme alanlarını tanımlamak için yararlı bir araçtır. Yalın prensipler iyi tanımlanmış ve üretimde geniş bir uygulanabilirliğie sahip olsa da bilgi teknolojisine olan yayılımı hala sınırlıdır. Bu çalışmada, bir firmanın BT projelerinin yönetimi değer odaklı yaklaşımla ele alınmış ve Değer Akış Haritalama yöntemi kullanılmıştır.

Her bir proje kümesi için mevcut durum değer akış haritası çizilmiştir. Mevcut durum değer akış haritasının ve gelecekteki durum değer akış haritasının analizi, kalite, maliyetler ve zamanın iyileştirilmesi için potansiyel alanlar hakkında bir fikir sağlamıştır. Mevcut durum haritası derhal müdahale edilmesi gereken yazılım geliştirmenin odak alanlarının ne olduğunu öne çıkarmıştır. Son kullanıcı ile iletişim sıklığının artırılması ve uzun kuyruk sürelerinin elenmesi ile sorunların çoğunun çözülebileceği görülmüştür.

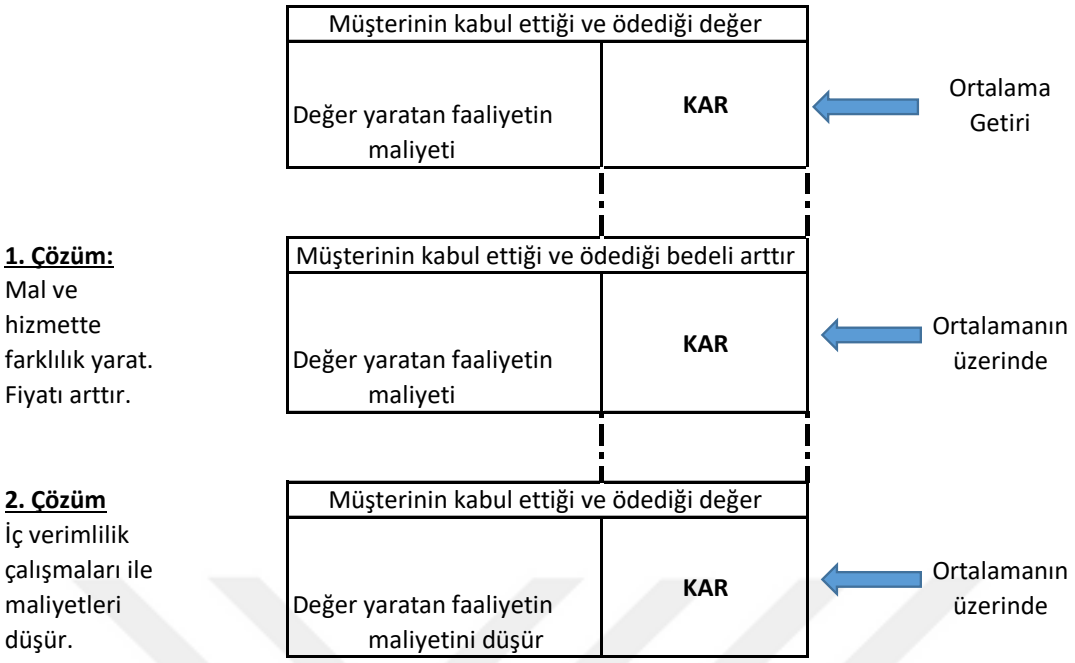
Çalışmanın akışı şu şekildedir: ikinci bölümde değerin tanımı, değerin yazılım projeleri için tanımı, yazılım geliştirme süreçlerinde yalın düşünme ve israfın yazılım projeleri için tanımı ile ilgili literatür araştırmasını sunulmuştur. Ardından üçüncü bölümde önerilen metodolojinin adımları ortaya konmuştur. 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. LİTERATÜR ARAŞTIRMASI

2.1.Değerin Tanımı

Değer, müşterinin kullanım durumlarında ulaşmasını ve amaçlarını gerçekleştirmesini kolaylaştıran (veya engelleyen) kullanımdan kaynaklanan ürün özelliklerinin, özellik performanslarının ve sonuçların değerlendirilmesi için müşterinin algılanan tercihidir (Hannan, 2011).

Bir hizmet veya mal ile değer yaratılıp yaratılmadığı müşterinin satın aldığı ürünü anlaması ile ortaya çıkar. Şekil 2.1'de de anlatıldığı gibi müşteri normal bir hizmet veya mal için ödediği ücret karşılığında sağladığı fayda arttıkça ya da satın aldığı hizmet veya mala daha az miktarda ücret ödedikçe değer yaratılmaktadır (Ülgen ve Mirze, 2007). Değer yaratan faaliyetlerin oluşturduğu toplam maliyet ve bu oluşturulan faaliyetler sonucu elde edilen toplam glirden oluşan iki temel unsur değeri belirlemektedir. Toplam gelir ile toplam maliyet arasındaki fark ise işletmenin karını oluşturmaktadır. Firmalar, karlarını arttırmak için rekabbette üstünlük elde etme çabası içerisindedirler. Eğer ortaya çıkan maliyetler, elde edilen toplam gelirden daha fazla ise o işletmede değer yaratmayan faaliyetler olduğu sonucuna ulaşılmaktadır. İşletmenin devamlılığını sürdürülebilmesi için değer yaratmayan faaliyetleri bırakması veya en azından azaltması gerekmektedir. Zaten elde edilen gelirin maliyetlerden yüksek olduğu durumlarda ise değer yaratan faaliyetlerin değerlerinin arttırılması için çabalanmalıdır. Şekil 2.1.'de görüleceği gibi maliyetlerin azaltılması ya da ürünün standartların dışına çıkarılması sonucu piyasa fiyatının üzerinde bir fiyat uygulanması ile değer arttırılabilmektedir. Değer müşteriler için yaratılmaktadır ve müşterinin ödemeyi göze aldığı fiyat bu değerin ölçüsünü oluşturmaktadır. Bir hizmet veya malın üretilmesi için oluşan maliyet, müşteri tarafından razı olunan fiyattan az olduğu sürece işletme karlı demektedir. Aksi halde o hizmet veya mal üretilmemelidir (Dingsoyr ve Lassenius, 2016)



Şekil 2.1. Değerin yaratılması**,** Bekçi ve Doğru (2011)'dan uyarlanmıştır.

Amaçlarına ulaşmak isteyen firmalar birbiri ile bağlatılı olan birçok faaliyette bulunurlar. İşletme hedeflerine ulaşmak için tüm birimlerdeki faaliyetler ve işletmenin sahip olduğu yetenek ve varlıklardan yararlanırlar. Rekabet üstünlüğü yaratmanın yolu birimlerin ve bu birimlerin faaliyetlerinin başarısından geçmektedir. Bu faaliyetler rakiplere göre daha fazla değer yarattığı sürece işletmeye rekabet üstünlüğü sağlamaya devam etmektedir (Ülgen ve Mirze, 2007).

Haksever ve diğ. (2004) değeri, hissedar, çalışan ve müşteri için finansal, finansal olmayan ve zaman ölçütlerine göre tanımlamışlardır.

Hissedar: Şirkette finansal payları vardır ve yatrırımların değer kazanmasını ya da gelir getirmesini bekler. Çalışan: Bir şirket için çalışan insanlar, geçimleri için aldıkları ücretlere veya maaşlara güvenirler. Ek olarak, sağlık sigortası, emekli aylığı, ücretli tatil süresi, aile hastalık izni, çocuk bakımı tesisleri vb. Gibi sosyal yardımlar, tazminat paketlerinin önemli bir bölümünü oluşturmaktadır.Müşteri: Müşteriler, satın aldıkları ürünlerden belirli faydalar beklemektedir. Genel olarak ihtiyaçları makul bir fiyata karşılandığı sürece şirkete sadık kalırlar. Müşteriler, ihtiyaç duydukları ürünleri başka bir şirketten uygun bir fiyatla elde edemezlerse, şirketin hayatta kalması ve refahı ile ilgilenebilirler.

Barier (1969) değeri, “bir kişiye veya tüzel kişiliğe olan bir ihtiyacı karşılama veya bir yarar sağlama gereksinimi olan malın, hizmetin veya faaliyetin kapasitesidir” şeklinde tanımlamıştır.

Porter (1985), değeri “alıcıların ne kadar ödemek istediği” olarak tanımlar ve bir firma, eşdeğer avantajlar için rakiplerden daha düşük fiyatlar teklif ettiğinde veya daha yüksek bir fiyatı dengeleyen benzersiz avantajlar sağladığında, bu üstün sonuç sağlar.

Çizelge 2.1: Paydaşlara göre değer tanımları, Haksever ve diğ. (2004)’den uyarlanmıştır.

Paydaş Grubu	Değer Boyutu	Yaratılan Değer
	Finansal:	Kar, gelir, hisse seneti değerlemesi. Karlı ve finansal açıdan stabil bir şirket.
Hissedar	Finansal olmayan:	Güvenilir gelir kaynağı, özerklik, iyi bir şöhrete 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 varolması
	Finansal:	Maaşlar, ek ödemeler, sağlık sigortası, emeklilik planı, ücretli izin süresi. İyi yönetilen, kar yapan ve finansal açıdan stabil bir şirkette çalışmak.
Çalışan	Finansal olmayan:	Güven, işbirlikçi bir iş çevresi, eğitim, iş güvenliği, yeni fırsatlar, esnek iş kuralları. Yönetimin çalışana duyduğu saygı, insan kaynaklarına yapılan yatırım, terfi imkanları
	Zaman:	Uzun vadede iş güvenliği, gelecekte sağlayacağı kariyer olanakları, bağlılığı ödüllendirme.
	Finansal:	Rekabetçi fiyatlarla iyi tasarlanmış üstün kaliteli ürünler. Güvenilir, az bakım gerektiren, düşük işletme maliyetine sahip ürünler.
Müşteri	Finansal olmayan:	Söz verildiği gibi çalışan, kurulumu ve kullanımı kolay olan ürünler. Nazik, güvenilir müşteri hizmetleri. Ürünün kullanımı, bakımı ve onarımı hakkında bilgi.
	Zaman:	Ürünün sağladığı zaman tasarrufu, uzun süre dayanan faydalar, vaat edilen zamanda veya müşterinin istediği zamanda teslim edilen ürünler. Arızalı ürünler için çok kısa duruş süresi, hızlı servis yanıtı ve onarım ile güvenilir ürünler. Arızalı parçaların hızlı değişimi.

Zeithaml (1988)'e algılanan değer, tüketicinin alınana veya verilene ilişkin algılara dayanarak bir ürünün faydasına ilişkin genel değerlendirmesidir. Alınan şeyler (miktar, yüksek kalite, rahatlık vb.) ve verilenler şeyler (harcanan para, zaman, çaba vb.) tüketiciden tüketiciye değişse de, değer al – ver ilişkisi arasındaki dengeyi temsil eder. Amazal ve diğ. (2014) değeri, müşterilerin mal veya hizmet alma masraflarından elde ettikleri kâr olarak tanımlar. Rahayu, hizmet tüketicileri tarafından alınan değerin doğrudan genel bir kullanıcı memnuniyeti hissine yol açabileceğini ekledi. Herdiyanti ve diğ. (2017), değerin iki bakış açısıyla, yani servis sağlayıcı ve servis tüketicisi tarafından anlaşılabileceğinin altını çizmektedir.

2.2.Değerin Yazılım Projeleri İçin Tanımı

Değere odaklanma, yazılım yoğun ürün geliştirme için kritik başarı faktörlerini araştıran, başarısız yazılım projelerinden başarılı olanları ayıran ve birincil kritik başarı faktörlerinin değer alanında olduğunu gösteren çalışmaların çoğuna paraleldir (Boehm, 2006). Bununla birlikte, değerin bir kavram olarak anlaşılması biraz sınırlıdır (Dingsoyr ve diğ., 2012; Dyba ve Dingsoyr, 2008; Racheva ve diğ., 2009).

Değer, geleneksel olarak kâr yaratma olarak görülür ve katma değer, bir yazılım iş perspektifinden dikkate alınması gereken bir maddi faaliyettir. Bununla birlikte, değer Khurum (2013) tarafından Yazılım Değer Haritalarında tarif edildiği gibi çok daha karmaşık bir kavramdır. Orada sadece geliştirme şirketleri için müşteri değeri ve finansal değerler hakkında değil, aynı zamanda yazılım geliştirme şirketi içindeki değer (veya varlıkların değerinin bozulması) ve miras gibi dahili değer unsurları üzerinde de dururlar.

Poppendieck (2011)'e göre yalın prensipleri değer üzerinde özel bir öneme sahiptir ve yalın yazılım geliştirmenin ilk prensibi, şirkete veya müşterilerine bir değer katmadığı sürece her faaliyet süreci boşa harcar.

Bununla birlikte, değeri nasıl tanımladığınız ve kullandığınızdan bağımsız olarak, yazılım yoğun ürün ve hizmetler geliştiren bir şirketin temel amacı, belirli bir yatırım için değer yaratma özelliğini en üst düzeye çıkarmaktır. Bunun mümkün olması için neyin değer olarak kabul edildiğini ve neyin değer yarattığını ve değer yaratıldığını sağlayan stratejilerin ne olduğunu anlamak gerekir (Aurum ve Wohlin, 2007).

Değerin üzerinde hemfikir olunan bir tanımı yoktur çünkü tüm işletmelerin iş değeri çok farklı yorumlara sahiptir. Değer kelimesinin genel kullanımı "önem veya fayda" dan "bir şeyin parasal değeri" ne kadar uzanır. Fayda ve parasal olarak tanımlandığında, yazılımın değeri geliştirme ekibi dışındaki paydaşlar tarafından belirlenir. Çevik gelişme ve yalın savucuları bir yazılım geliştirme projesi sırasında geliştirme ekibinin dış paydaşların değerinin ne olduğunu öğrenmeleri gerektiğini savunurlarken, geleneksel yaklaşım değeri kendi bakış açılarına göre tanımlamaları gerektiğini savurular (Patterson, 1989)

Yazılım değer boyutları literatürde aşağıdaki gibi sıralanmıştır.

Doğruluk: Bir yazılım geliştirmeden önce analist ve proje paydaşları ile bir araya gelerek gereksinimler doğrultusunda bir spefikasyon hazırlanır. Hazırlanan yazılımın oluşturulan spefikasyona sadık kalınması bir değer tanımıdır. Bu bir ölçüde yazılımın doğru çalışıp çalışmadığının da ölçütüdür. (Ebert ve Dumke, 2007)

Güvenilirlik: Yazılım projesinin uygulamaya geçtikten sonraki süreç içerisinde yani canlı ortamda ne sıklıkla bir hata meydana geleceğidir. (Ebert ve Dumke, 2007)

Veri Bütünlüğü ve Güvenliği: Büyük programlar veya program sistemleri genellikle kalıcı bir veri kütlesini sağlama sorumluluğuna sahiptir. Verilerin tutarlılık sınırlamalarını sağlaması gerekebilir. Öyleyse, yazılım bu kısıtlamaları ihlal etmediğinden emin olmalıdır. Ek olarak, verilere erişimi sınırlamak yazılımın sorumluluğunda olabilir. Verilerin değişikliklerini sınırlamak özellikle önemlidir. Ve ciddi sonuçları olan değişiklikler çok kolay olmamalıdır. (West ve Hommond, 2010)

Teslim zamanı: Zamanında teslimat - Yazılımın alıcısı her zaman ürünün teslimatının zamanında yapılmasını bekler. Alıcı bir işletme firması olduğunda, kendi taahhütlerini yerine getirmek için yazılıma verilen tarihte ihtiyaç duyar. Yazılımın son teslim tarihinden önce teslim edilememesi durumunda, alıcı ürünü teslim almaktan vazgeçerek, başka bir firmaya yönelebilir. Bu yüzden yazılımın son teslim tarihinden sonra teslim edilmesi yerine bazen hatalı şekilde teslim edilmesi tercih edilebilmektedir. (Jansen ve Saiedian, 2005)

Minimum maliyet: Alıcı, istenen işlevselliği ve kaliteyi elde edecek en düşük fiyatı ödemek isteyecektir. (Jansen ve Saiedian, 2005)

Kullanım kolaylığı: Kullanıcıya anlaşılır bir arayüz sunması ve uygulamayı kullanılabilirliğinin kolay olması (West ve Hommond, 2010)

Geliştirilmeye açık olması: Yapılan yazılım üzerine konan ek istekleri karşılayabiliyor olmalıdır. (West ve Hommond, 2010)

2.3.Yazılım Geliştirme Süreçlerinde Değer Akış Haritalama

Yalın yazılım geliştirme, geliştirme zamanını ve kaynaklarını optimize etmeye, atıkları ortadan kaldırmaya ve nihayetinde yalnızca ürüne değer katan unsurları sağlamaya dayanan çevik bir çerçevedir (Poppendieck ve Poppendieck, 2003). Yadav ve diğ. (2019) yazılım geliştirme projelerinde yalının 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). Önerilen yalın araçlardan biri de Değer akış haritalamadır. Değer akış haritalama yönteminde süreçteki her bir faaliyet analiz edilir ve bu faaliyetlerin sürece değer katıp katmadığı sorgulanır. Bir süreci oluşturan her adım değer katan veya değer katmayan faaliyet olarak sınıflandırılır.

Bu çalışmada yalın düşünme prensiplerinin yazılım geliştirme sürecine uyarlanması ele alınmış ve değer akış haritalama metoduna odaklanılmıştır. Yazılım geliştirmeye değer akış haritalama metodunun uygulanması konusunda az sayıda yayın bulunmaktadır. Scopus'a veri tabanında (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 7 çalışma (makale ve bildiri) listelenmiştir.

Süreçlerdeki sorunları göstermek / vurgulamak, genellikle yalın yazılım geliştirmede kullanılır ve bariz faydalara rağmen, göstergelerin yanlış kullanımı sonucu oluşabilecek tehlikeler vardır. Örneğin; yetersiz göstergeler kullanmak, paydaşları hatalı kararlara doğru yönlendirebilir. ,Staron ve diğ. (2013) ölçüm sistemleri (yani hem önlemler hem de göstergeler) tarafından sağlanan bilgilerin eksiksizliğini değerlendirmek için bir yöntem sunmuşlardır. Bu yöntem, Ericsson birimlerinden

birinde paydaşlara yazılım kalitesiyle ilgili gelecek sorunlar hakkında erken bir uyarı sistemi sağlamak için uygulanmıştır.

Anand ve diğ. (2014) lojistik tedarikçilerine tedarik zinciri yazılım çözümleri sağlayan bir Hint yazılım firmasının iş sürecini yeniden yapılandırmak için farklı yalın araçlar sunarak VSM uygulaması ile çeşitli atıkları tespit etmişlerdir.

Ali ve diğ. (2015) VSM'yi yazılım süreci simülasyon modellemesi ile birleştirerek eksikliklerinin üstesinden gelmek ve VSM'yi simülasyon ile yürütme süreci hakkında yansımalar sağlamak için Ericsson AB'nin İsveç'teki iki ürünü için VSM'yi kullandılar.

Yazarların 2016 yılında yürüttükleri çalışma ise İsveç AB Ericsson'daki büyük bir ürün için akış ve değer akış haritalama kombinasyonu, sistematik olarak sorunların ortaya çıkarılmasında ve çözümlerinin karakterize edilmesinde başarılı olmuştur; bu da bilgi akışı ile ilgili konulara odaklanarak israf giderimi için pratik yararlılıklarını göstermektedir. (Ali ve diğ., 2016)

Softtech A.Ş. firmasında konu kayıt çözümü süreci "değer akışı haritalama" metodolojisi ile incelenmiş, ilgili veriler ölçülerek iyileştirme amacıyla analiz edilmiş, elde edilen iyileştirmeler ve önümüzdeki dönemde yapılması planlanan diğer araştırma çalışmaları paylaşılmıştır (Aktaş ve diğ., 2018).

Özcan ve McCaffery (2018) hibrit değerlendirme yaklaşımı, MDevSPICE tabanlı süreç değerlendirme yöntemini, değer akışı eşlemesi yoluyla iyileştirme ihtiyaçlarının önceliklendirilmesi ve KATA tekniği kullanılarak süreç iyileştirmenin sağlanması için adımlarıyla birleştirmiştir. Bu yaklaşım, yasal standartların gerekliliklerini yerine getirirken çevik yöntemleri tıbbi cihaz yazılımı geliştirme sürecine entegre etmiştir.

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

Tüm makaleler de ortak olarak yazılım geliştirme alanında çevik yöntemler ve değer akış haritalama yöntemi kullanılarak israflar belirlenmiş. Böylece israflar ortadan

kaldırılabilmesi için analizler ile süreç iyileştirmeleri yapılabilmiştir. Bizim çalışmamızda da bu makaleler referans alınarak değer akış haritalama metodu kullanışmış ve israflar belirlenerek süreçte çevrim süresini kısaltacak iyileştirme önerileri sunulmuştur.

2.4.İsrafın Yazılım Projeleri İçin 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; Alahyari ve diğ., 2017).

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 (evrak gibi) 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 gerekli olandan fazla özellik koymak aslında ciddi bir israf demektir.
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 atıklardan biridir. Projenin başlamasındaki gecikmeler, tekrar gözden geçirmeler, onay bekleme, test ve canlı ortama geçişteki gecikmelerden oluşmaktadır.
6. Hareket (motion): Yazılım geliştirme sürecindeki çalışanların bir sorun veya gereksinim için harekete geçmesi ile ilgilidir. Teknik bir sorunun çözümü, müşteri veya temsilcilere erişim, bir testin sonuçlarını görme ihtiyacı gibi.

7. Hatalar (defects): Süreçteki en gözle görülebilen atık kaynağıdır. Hatanın israf miktarı hatanın etkisine ve tespit edildiği süreye bağlı olarak ortaya çıkmaktadır.

Senado ve diğ. (2017) bir yazılım geliştirme danışmanlık firmasında sekiz yazılım geliştirme projesinin iki yıllık beş aylık bir katılımcı-gözlem çalışması ve 33 yazılım mühendisiyle, tasarımcılarıyla ve ürün yöneticileriyle röportaj yaparak, yazılım geliştirme projeleri için dokuz tür israf ortaya koymuşlardır: yanlış özellik veya ürün oluşturma, iş yükünü yanlış yönetme, yeniden çalışma, gereksiz karmaşık çözümler, dışsal bilişsel yük, psikolojik sıkıntı, bekleme / çoklu görev, bilgi kaybı ve etkisiz iletişim. (Sedano ve diğ., 2017)

Alahyari ve diğ., (2017) çalışmalarında bu israfları önceliklendirmiş ve aralarındaki ilişkileri ortaya koymuşlardır. Buna göre görev değiştirme, hatalar ve bekleme yazılım projelerinde karşılaşılan israfların başında gelmektedir.

Bu çalışmada Haksever ve diğ.. (2004) tarafından ortaya konduğu gibi, değeri sadece müşteri ve finansal değerler için değil, yazılım geliştirme içerisindeki diğer değer unsurları için de tanımlayan yaklaşım dikkate alınmıştır. Müşteri açısından teslim zamanı, doğruluk, kullanım kolaylığı, geliştirilebilirliğe açık olması, minimum maliyet tanımları yapılırken, çalışan için veri bütünlüğü ve güvenliği, güvenilirlik ve zaman açısından değerlendirildiğinde minimum maliyet tanımları yapılabilmektedir. Hissedar açısından ise teslim zamanı, minimum maliyet, doğruluk kriterleri değeri tanımlamaktdır. Poppendieck (2011)'in yalın prensibinden yola çıkarak şirkete veya müşterilerine değer katmadığı sürece her faaliyetin süreci boşa harcadığını göz önünde bulundurulmuş ve israfları ortadan kaldıracak öneriler sunulmuştur.

3. METODOLOJİ

Bu çalışmanın yöntemsel temelini oluşturan yalın ilkeler Womack ve Jones (2003) tarafından aşağıdaki şekilde ortaya konmuştur;

1. Değeri tanımla. (ürün - müşteri ve süreç odaklı)
2. Değer akışını haritala ve değer yaratmayan adımları (israfları) belirle.
3. İsrafı azalt, akışı sağla.
4. Değer odaklı yeni akış önerisi geliştir.
5. Sürekli mükemmeli ara.

3.1. Metodolojinin Adımları

Yalın ilkeler çerçevesinde yazılım projelerinin değer odaklı analizi ve iyileştirilmesi için izlenmesi önerilen adımlar şöyle tanımlamıştır (Şekil 3.1):

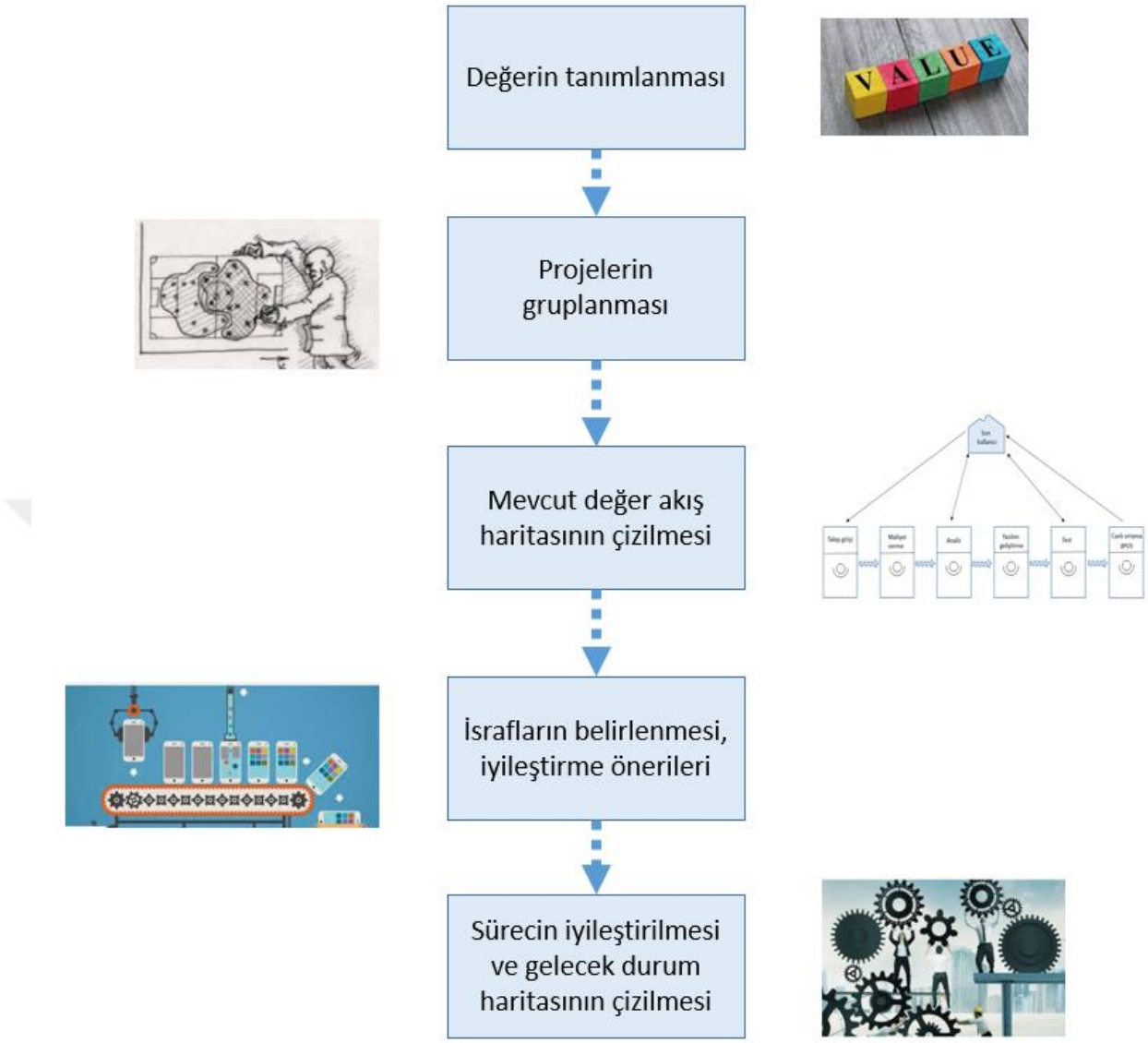
1. Müşteri, ürünler, işletme süreçleri ve diğer paydaşlar dikkate alınarak yazılım projeleri için değer tanımının yapılması ve değer boyutlarının (kriterlerinin) belirlenmesi. Bu aşamada literatürde yazılım geliştirme ile ilgili değer tanımlarından yola çıkan ve yazılım sağlayıcının mevcut durum ve beklentileri 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 1. Adımda 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. İki adımlı kümeleme analizi ile ilgili bilgi 3.1 başlığı altında verilmiştir. 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 odaklı yalın prensiplerin yazılım geliştirmeye uyarlanmasında kullanılması önerilen araçlardan biri de değer akışı haritalarıdır (Poppendieck ve Poppendieck, 2003). 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, 1999). Bir süreci oluşturan her adım değer katan veya değer katmayan faaliyet olarak sınıflandırılır. Yazılım projeleri için değer akış haritalama konusunda az sayıda çalışma bulunmaktadır (Mujtaba ve diğ. 2010; Kasoju ve diğ., 2013; Khurum ve diğ., 2014; Ali ve diğ., 2015; Ali ve diğ. 2016; Yadav ve diğ., 2018; Yadav ve diğ., 2019). 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., (2017) tarafından listelenmiştir. Mevcut süreç, değer tanımları çerçevesinde değer yaratan ve yaratmayan aktiviteler olarak irdelenip israflar ortaya konur. İsrafların ortadan kaldırılmasına veya azaltılmasına yönelik iyileştrme 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, 1999).



Şekil 3.1: Uygulama adımları akış diyagramı

3.2.İki Adımlı Kümeleme Analizi

Kümeleme, verileri birbirine "benzer" ve birbirinden "farklı" veri noktaları içeren kümeler halinde gruplandırmak için kullanılan bir makine öğrenme tekniğidir. İki adımlı kümeleme analizi klasik kümeleme analizi yöntemleriyle karşılaştırıldığında, hem sürekli hem de kategorik özelliklere olanak sağlar. (Ahmad ve Khan 2017) İki aşamalı kümelemenin bir diğer önemli avantajı, yöntemin istatistiksel kriterlere göre küme sayısını otomatik olarak seçmesine izin vermesidir. Bunu yaparken de iki aşamalı bir yaklaşım izler (Bacher ve diğ. 2004).

IBM SPSS iki adımlı kümeleme analizi, Banfield ve Raftery (1993) tarafından kullanılan model tabanlı mesafe ölçüsünü hem sürekli hem de kategorik değişkenleri içerecek şekilde ele alır. BIRCH'a benzer iki aşamalı kümeleme yaklaşımını kullanır (Zhang ve diğ., 1996) ve optimum küme sayısının otomatik olarak bulunmasını sağlar. (IBM SPSS, n.d.)

İlk aşamada yöntem, araştırmacının belirtimlerine bağlı olarak Akaike'ın Bilgi Kriterine (AIC) veya Bayes Bilgi Kriterine (BIC) dayalı olarak maksimum küme sayısını belirler. Bu kriterler, analizden kaynaklanan log olasılık değerine kümeler sayısıyla ifade edilen çözümün karmaşıklığını cezalandıran farklı terimler ekler - daha fazla kümeye sahip çözümler daha güçlü bir ceza süresi gerektirir. SPSS'de, maksimum küme sayısı, k kümeli bir çözüm için AIC (veya BIC) ve tek kümeli bir çözüm arasındaki oranla belirlenir. Bu oranın program tarafından üstlenilen belirli bir eşik değerden daha küçük olduğu çözüm, maksimum küme sayısıdır.

İkinci aşamada, girdi olarak AIC (veya BIC) değerlerini kullanarak farklı küme çözümleri arasındaki mesafelerin oranını hesaplar. Ortaya çıkan oran, çıkarılacak kümelerin nihai sayısını belirler.

İki adımlı kümeleme analizi sonucunda kaç küme oluşacağı sorusunun cevabı veri setine bağlıdır. K-means ve EM yönteminin diziyi oluşturmak için birden çok kez çalışması gerekir (her bir küme sayısı içi bir kere). Küme sayısını otomatik olarak belirlemek için SPSS hiyerarşik kümeleme yöntemiyle iyi çalışan iki aşamalı bir prosedür geliştirmiştir. İlk adım, belirli bir aralıktaki küme sayısı için Denklem(1)'deki şekilde BIC'yi hesaplar ve bu değeri küme sayısı için başlangıç tahmininde kullanır. İkinci adım, her bir hiyerarşik kümeleme aşamasında en yakın iki küme arasındaki mesafedeki en büyük değişikliği bularak ilk tahminde değişiklik yapar.

$$BIC_i = -2 \ln (L_i) + k_i \ln(n) \quad (1)$$

n : örnek sayısı

k_i: i modelindeki parametre sayısı

$ln\ (L_i)$: tahmini model i için olabilirlik fonksiyonunun maksimize edilmiş değeridir.

Literatürde yazılım projelerinin kümelenmesi ile ilgili çeşitli çalışmalar vardır. Barki vd. (1993), 23 yazılım riskini faktör analizi kullanarak beş grupta toplamışlardır. Tek

kültürlü bir bakış açısının önyargısını azaltmak için, Schmidt ve diğ., (1998) Hong Kong, Finlandiya ve Amerika Birleşik Devletleri'nden uzmanların seçeneklerini entegre etmek için Delphi tekniği kullanmışlardır. 14 türe ayrılmış 53 risk maddesi belirleyip ve kültürel farkın listeyi etkileyebileceğini ve kültürler arası bir bakış açısıyla sadece 11 yazılım riskinin uygulanabileceğini iddia etmişlerdir.

Huang ve Han (2008) iki adımlı kümeleme analizi yönetimini kullanarak yazılım projelerinin risk faktörlerini sınıflandırmış ve proje yöneticilerinin daha yüksek başarı oranlarına yol açabilecek önemli riskleri nasıl hafifleteceğine veya ortadan kaldıracağına dair karar vermeleri için önerilerde bulunmuştur. Mevcut çalışmalar proje riski odaklıdır. Bu çalışmada değer odaklı bir kümeleme yapılmıştır. Analize konu olan proje özellikleri karma veri özellikleri taşıdığı için iki adımlı kümeleme analizi (Two Step Cluster Analysis) kullanılmıştır.

Uygulamada, yazılım projelerini değerlendirmek için hem kategorik hem de metrik verilere sahip olduğumuz için projeleri kümelemek için IBM SPSS 2 Adımlı Kümeleme Analizi (TwoStep Cluster Analysis) tekniği kullanılmıştır.

4. UYGULAMA

Bu çalışmada İstanbul'daki bir finans firmasının Bilgi Teknolojileri departmanının yürüttüğü yazılım geliştirme projeleri kullanılmıştır. Yazılım geliştirme projelerinde 32 analist ve 29 yazılımcı olmak üzere toplam 61 personel görev almakta ve yazılım geliştirme metodu olarak ta Klasik (Şelale) yaklaşımı kullanılmaktadır.

4.1.Değer Tanımı ve Proje değerlendirme Kriterlerinin Belirlenmesi

Literatür taraması bölümünde aktarılan değer tanımlarından yola çıkarak yazılım projeleri için Çizelge 4.1'de yer alan kriterler ortaya konmuştur. Kriterlerin açıklamaları aşağıda detaylandırılmıştır.

Maliyet: Çalışmamızda toplam proje süresini ifade etmektedir. Bir iş için harcanması planlanan minimum süre maliyet olarak nitelendirilmektedir. Müşteriye bu süre iletilir. Günlük 8 saatten hesaplanarak gün cinsinden bildirilir.

Efor: İşin tamamlanması için harcanan süreyi ifade etmektedir. Teslim zamanı harcanan eforların toplamı ile hesaplanabilmektedir. Maliyet tanımında olduğu gibi 1 günün 8 saat olduğu farzedilerek 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 geliştirilebilirliğe açık olduğu anlamına gelmektedir. Nümerik olarak ifade edilir. Her yeni istek için bir ek talep kartı açılır.

Güncelleme Sayısı: Proje yürütülürken meydana gelen hatalar için yapılan güncelleme sayısıdır. Doğruluk ve güvenilirlik değer tanımlarını ifade etmektedir. Ne sıklıkla bir hata geleceğini ve hazırlanan spesifikasyon ile çalışmanın birbirinden ne kadar farklı olduğu güncelleme sayısı ile orantılıdır.

Tecrübe yıl sayısı: Projede çalışanların toplam deneyim yılı sayısını ifade etmektedir.

Çizelge 4.1 : Sınıflandırmada kullanılan bağımsız değişkenler

Kriterler	Kısaltma	Veri Tipi	Kodlama
Tahmin edilen süre	Maliyet	Nümerik(gün)	-
Harcanan efor	Efor	Nümerik(gün)	-
Projenin önceliği var mı?	Öncelik	İkili Kategorik	1-0
Projenin amacı	Amaç	Kategorik	1…5
Projede çalışan kişi sayısı	Kaynak	Nümerik(adam)	-
Ek talep sayısı	Ek Talep	Nümerik	-
Güncelleme sayısı	Güncelleme	Nümerik	-
Projede çalışanların tecrübe yılı toplamı	Tecrübe	Nümerik(yıl)	-

4.2.Yazılım Projelerinin Gruplanması

Uygulamada, yazılım projelerini değerlendirmek için hem kategorik hem de metrik verilere sahip olduğumuz için projeleri kümelemek için IBM SPSS 2 Adımlı Kümeleme Analizi (TwoStep Cluster Analysis) tekniği kullanılmıştır. 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 analizinin normallik ve multinomiallik varsayımlarını karşılamamasına rağmen yöntem varsayımların ihlaline karşı duyarsız (robust) olduğu için (Norusis, 2011) 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. Değişkenlerarası en güçlü korelasyon Çizelge 4.2 de görüleceği 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. Kümeleme kriteri olarak AIC ve BIC kullanarak ayrı ayrı yapılan analizler projeleri benzer şekilde 4 kümeye ayırmıştır. İki Adımlı Kümeleme analizi sonuçları Çizelge 4.3. ve Çizelge 4.4'de görülmektedir.

Çizelge 4.2 : Değişkenler arası korelasyon matrisi

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

Çizelge 4.3 : İki adımlı kümeleme sürecinin sonuçları

Küme Sayısı	Schwarz's Bayesian Criterion (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	,526	1,758
4	949,523	-9,259	,060	1,663
5	978,091	28,567	-,185	1,148
6	1014,033	35,943	-,233	1,456
7	1065,548	51,514	-,334	1,033
8	1118,157	52,609	-,341	1,055
9	1172,476	54,319	-,352	1,191
10	1231,827	59,351	-,384	1,015
11	1291,570	59,744	-,387	1,463
12	1359,501	67,931	-,440	1,006
13	1427,540	68,039	-,441	1,183
14	1498,305	70,765	-,458	1,130
15	1570,779	72,475	-,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.

Çizelge 4.4 : Projelerin kümelere dağılımı

		N	% of Combined	% of Total
Cluster	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%

Çizelge 4.5, tüm projelerin ve her bir yazılım projesi kümesinin profilini özetlemektedir. Metrik değişkenler olan maliyet, efor, kaynak, ektalep, 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 görülebilmektedir. Şekil 4.1'te değişkenlerin her bir küme içerisinde önem bazında sıralanışları yer almaktadır.

Tek tek değişken bazında incelediğimizde yüksek maliyete ve efora sahip projeler Küme-3'te toplanmıştır. Küme-1'de kaynak sayısı da diğer kümelere göre daha yüksektir. Ama diğer kümelerde 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:

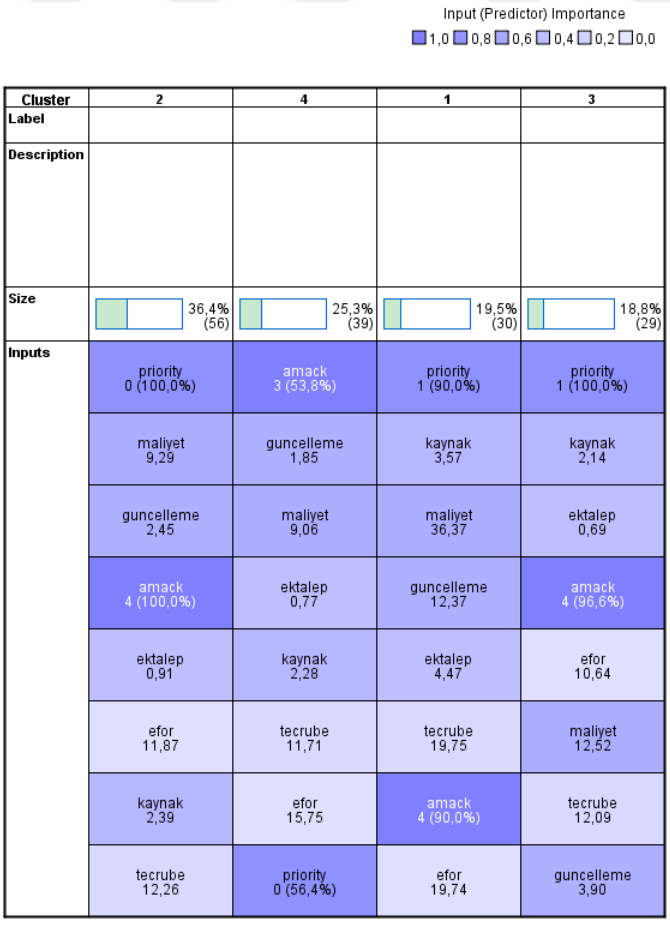
- Proje maliyeti ve kaynak gereksinimi yüksek.
- Daha fazla efor gerektiren.
- Ek talep sayısı ve hata miktarı (güncelleme sayısı) yüksek.
- Daha tecrübeli personelin çalıştığı projeler.

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

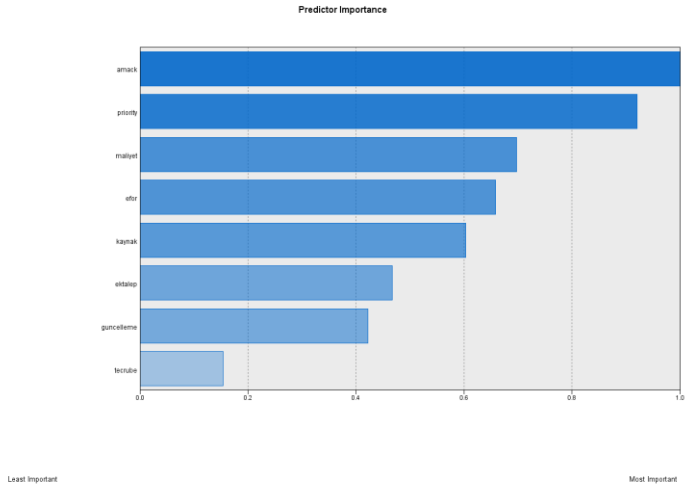
Çizelge 4.5 : Projelerin özet profilleri

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
Ek Talep	1,53	4,46	0,92	0,69	0,77
Güncelleme	4,50	12,37	2,45	3,90	1,85
Tecrübe Yılı	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 iyileştirme	111	27	56	28	0
Yasal zorunluluk	20	2	0	0	18

Şekil 4.1 : Değişkenlerin her bir küme içerisinde önem sırası



Şekil 4.2’ de kümeleme yapılırken değişkenlerin önem sıralamaları yer almaktadır. En büyük öneme sahip değişken olan amaç, hissedar için önemli bir kriterdir. Amacı öncelik takip etmektedir. 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 önemi 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 geliştirilebilirliğe ne kadar açık olduğunu da göstermektedir. Bu bakımdan müşteri için büyük bir öneme sahiptir. Kaynak en az önemli üçüncü değişken olarak hissedarı doğrudan etkilemektedir. Çalışan açısından da ne kadar fazla kaynak o kadar iş olanağı anlamına geldiği için çalışan açısından da dolaylı etkisi olan 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.



Şekil 4.2 : Kümeleme yapılırken değişkenlerin önem sıralamaları

4.3.Mevcut Değer Akış Haritalama

Her bir kümede yer alan süreçlere ait adımlar için veriler, proje bilgilerinin depolandığı Jira ortamından çekilerek ortalama işlem ve bekleme süreleri hesaplanmıştır. Yazılım geliştirme sürecindeki adımlar şu şekildedir.

1. Yazılım geliştirme sürecindeki ilk adım son kullanıcının talep uygulaması üzerinden talebi girmesi ile başlar.
2. Yazılım geliştirme sürecinin ne kadar süreceğinin hesaplanmasıdır. Maliyetlendirme süreci Şekil 4.3'te görüldüğü gibi 4 aşamadan oluşur. Öncelikle analist kendi maliyetini belirler ve ardından yazılımcı kendi maliyetini belirler.
3. Analiz süreci, son kullanıcının talebi üzerine gereksinimler belirlenerek bir analiz dökümanı hazırlanır. Şekil 4.3'deki gibi döküman son kullanıcının da onayından geçer ve eklemeler / düzenlemeler sonucu, fonksiyonların da eklenmesi ile döküman yazılıma iletilir.
4. Şekil 4.3'te yazılım geliştirme sürecinin adımları yer almaktadır.
5. Şekil 4.3'te 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. Kullanıcı testinin de tamamlanması ile Şekil 4.3'te adımları yer alan canlı ortama geçiş süreci ile proje tamamlanmış olur.

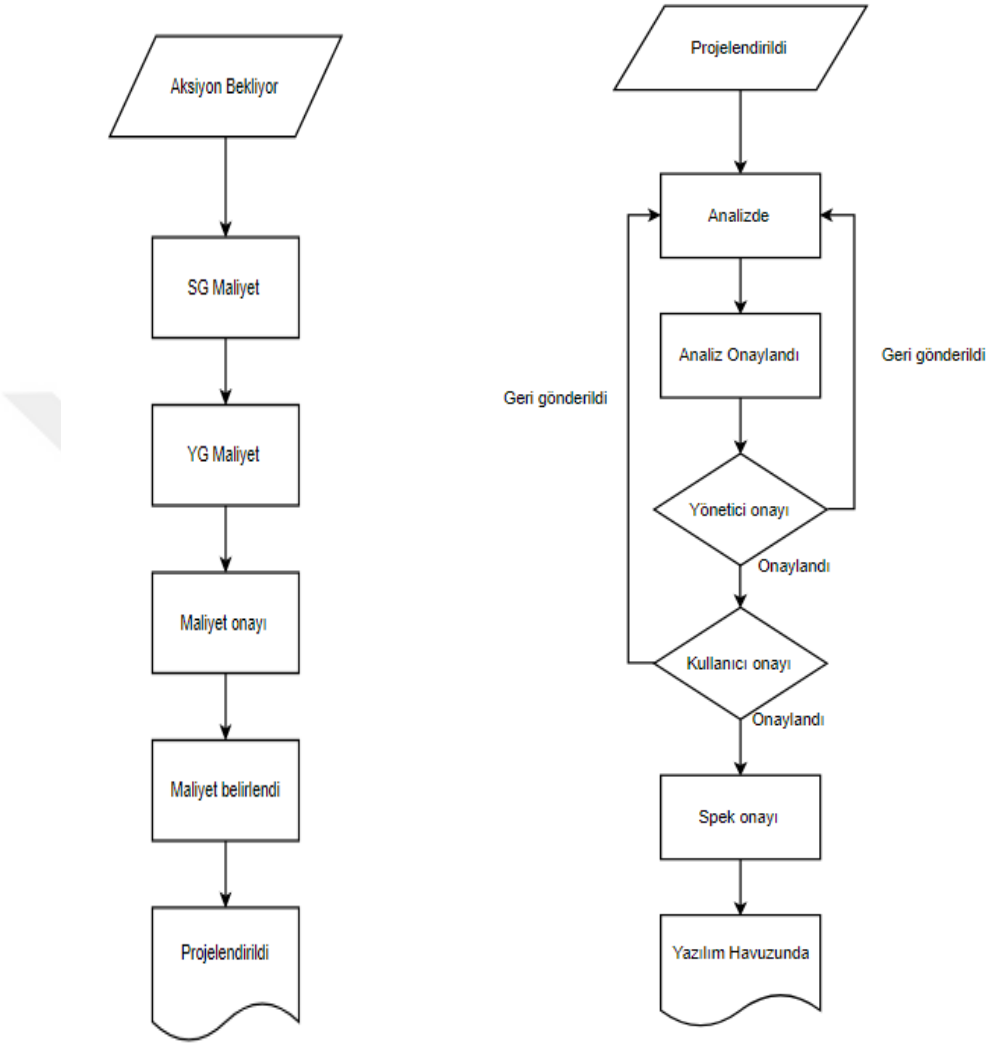
Şekil 4.6'da birinci kümede 175,2 günlük çevrim süresinin 27,5 günün işlem süresine ait olduğu görülmektedir. Şekil 4.7'deki ikinci kümede 25,3 gün işlem süresine karşın 141,5 gün çevrim süresi bulunmaktadır. Şekil 4.8' de mevcut akış haritası çizilen üçüncü sınıf küme en fazla işlem ve çevrim süresine sahip sınıftır. 217,3 gün çevrim süresinin sadece 52,17 günü işlem süresine aittir. Şekil 4.9'da yer alan dördüncü küme ise en az çevrim süresine sahip kümedir: 14,4 gün işlem süresi, 85,8 günlük çevrim

süresi bulunmaktadır. Genel olarak tüm kümelerin işlem süresine karşılık bekleme süreleri dolayısıyla çevrim süreleri oldukça fazladır.

Mevcut durum değer akışları ve süreler incelenerek israflara sebep olan faktörler bulunup elendikten sonra gelecek durum değer akış diyagramı çizilmiştir. Çalışmamızda yazılım süresini, maliyetlerini ve yeniden çalışmayı azaltmak için bir yazılım geliştirme süreci için değer akış haritalamasını kullanacağız. Yazılım geliştirme süreci akışını, her bir adımın ne kadar sürede tamamlanacağını, her adım arasındaki bekleme süresini ve mevcut durum değer akış haritasını çizerek başlar. Daha sonra tüm süreçteki darboğazlar tespit edilir, döngü sürelerini, bekleme sürelerini azaltmak için öneriler geliştirilir ve gelecekteki ayrıntılı bir durum değeri akış haritası geliştirilir.

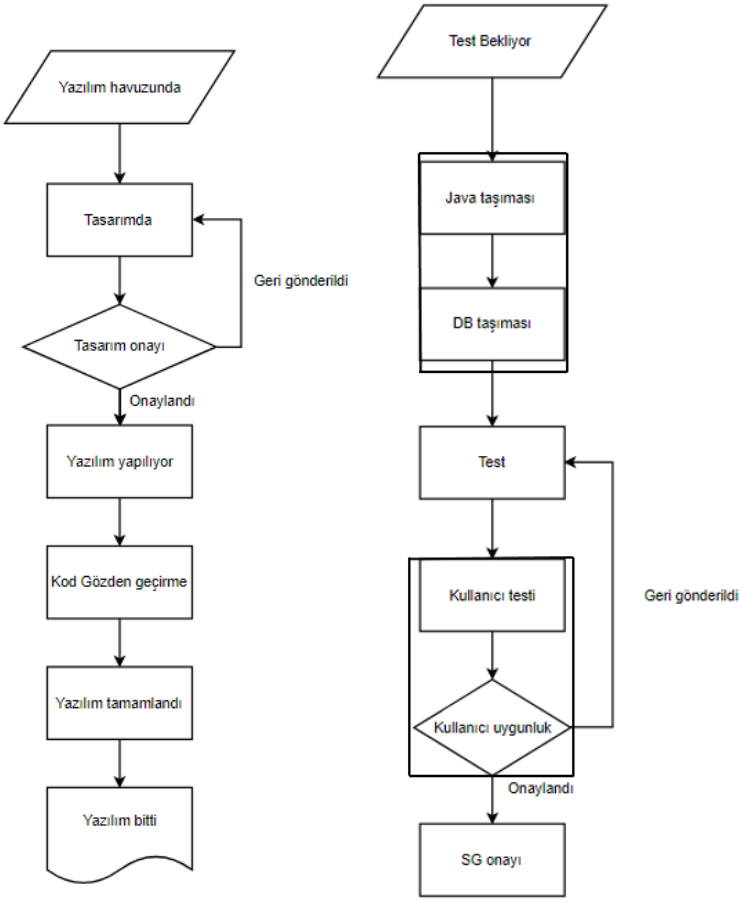
Çizelge 4.6 : Süreçlere ait ortalama süreler

süre(gün)	Küme-1	Küme-2	Küme-3	Küme-4	Ortalama
Maliyetlendirme	0,97	0,51	0,52	0,52	
SG maliyet	0,05	0,16	0,04	0,04	0,0725
YG maliyeti	0,52	0,1	0,18	0,18	0,245
Maliyet onaylama	0,24	0,12	0,18	0,18	0,18
Maliyet belirleme	0,16	0,13	0,12	0,12	0,1325
Analiz	2,94	1,48	1,95	1,96	2,083
Analizde	1,96	0,98	1,58	1,48	1,5
Analiz onay	0,01	0,04	0,15	0,11	0,0775
Yönetici onay	0,6	0,32	0,13	0,2	0,3125
Kullanıcı onay	0,26	0,1	0,05	0,13	0,135
Spek onay	0,11	0,04	0,04	0,04	0,0575
Yazılım Geliştirme	8,92	5,93	6,54	7,29	7,17
Tasarımda	7,8	4,28	5,15	6,1	5,832
Tasarım onay	0,04	0,1	0,01	0,01	0,04
Yazılım yapılıyor	0,98	0,69	1,1	0,34	0,7775
Kod gözden geçirme	0,04	0,28	0,14	0,24	0,175
Yazılım tamamlandı	0,06	0,58	0,14	0,6	0,345
Test	5,62	3,55	1,51	4,68	3,84
Java-DB taşıması	1,89	0,32	0,5	0,91	0,905
Test	3,35	2,83	0,89	3,54	2,653
Kullanıcı onay	0,23	0,39	0,09	0,22	0,2325
SG onay	0,15	0,01	0,03	0,01	0,05
Canlı Ortama Geçiş	0,26	0,04	0,05	0,02	0,0925
Taşıma formu girişi	0,25	0,01	0,04	0,01	0,0775
Taşındı	0,01	0,02	0,01	0,01	0,0125

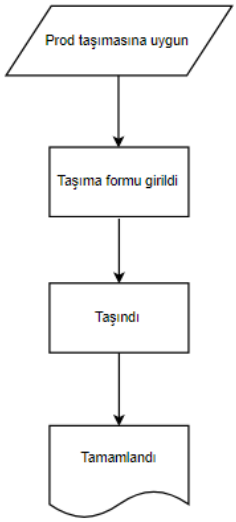


Şekil 4.3 : Maliyetlendirme ve Analiz süreçleri

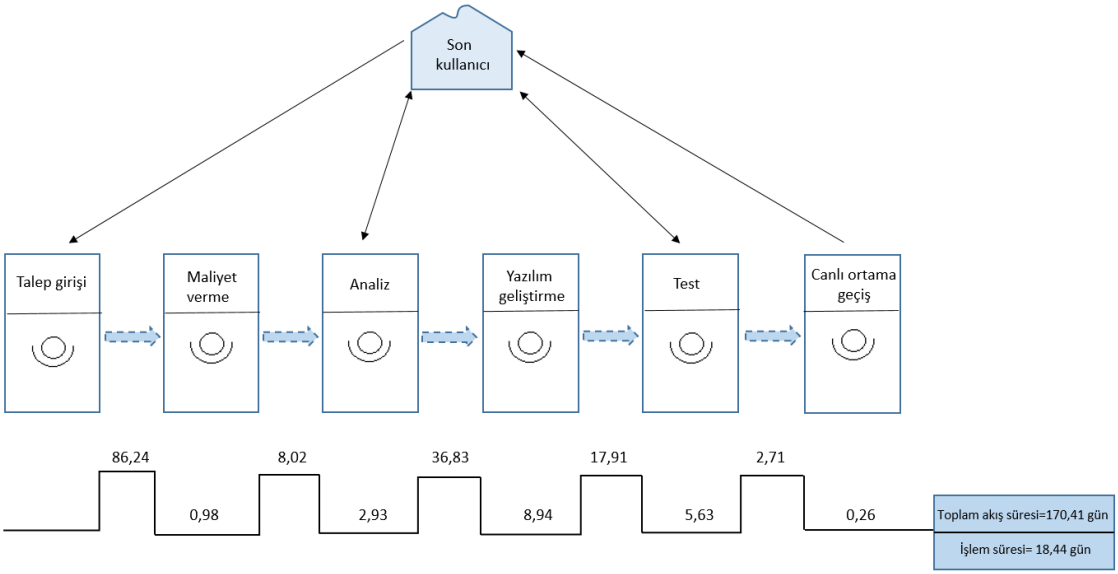
Her bir kümeye ait süreçler, bu süreçlerin ortalama süreleri ve toplam işlem ve çevrim süreleri Şekil 4.6, Şekil 4.7, Şekil 4.8 ve Şekil 4.9'da gösterilmiştir. Son kullanıcı ile başlayan süreç, son kullanıcıya doğru ve sağa doğru değişken bir akış göstermektedir.



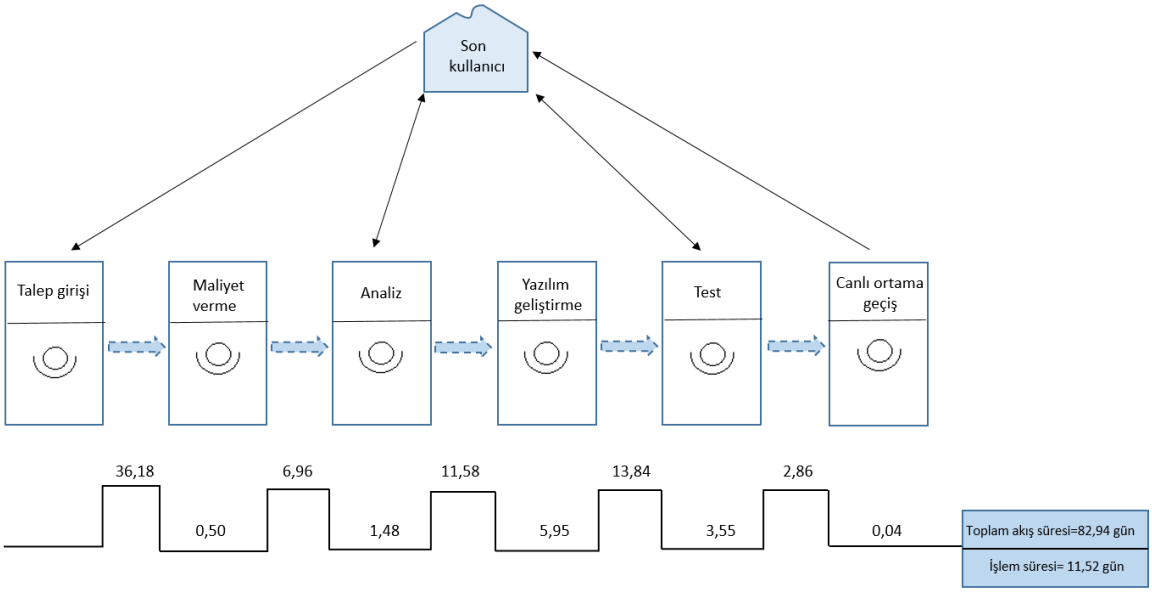
Şekil 4.4 : Yazılım geliştirme ve Test süreçleri



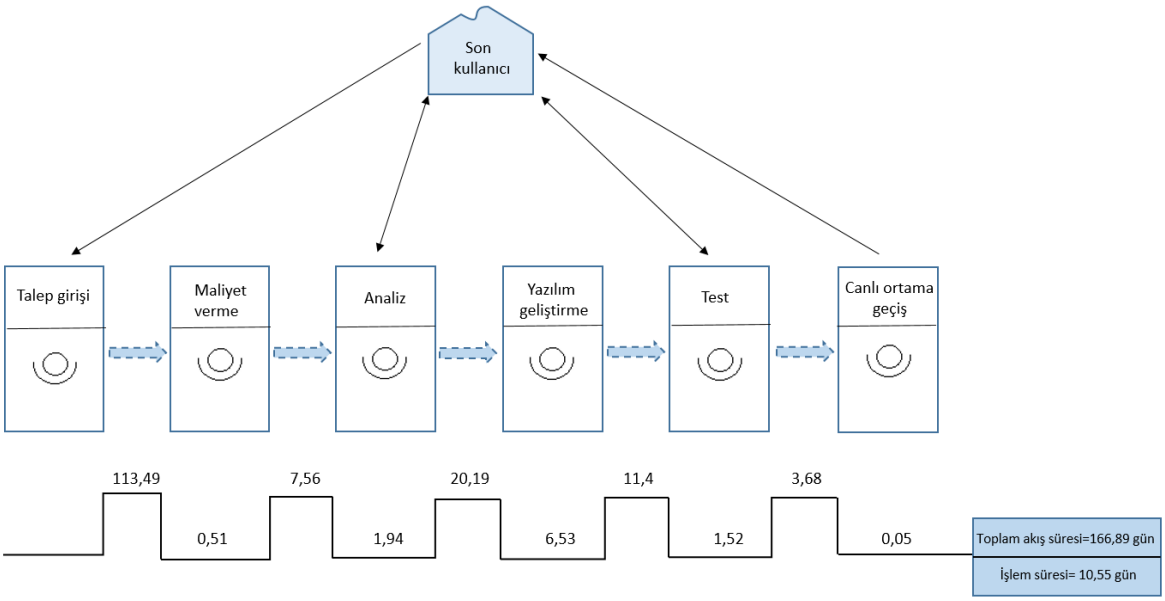
Şekil 4.5 : Taşıma süreci



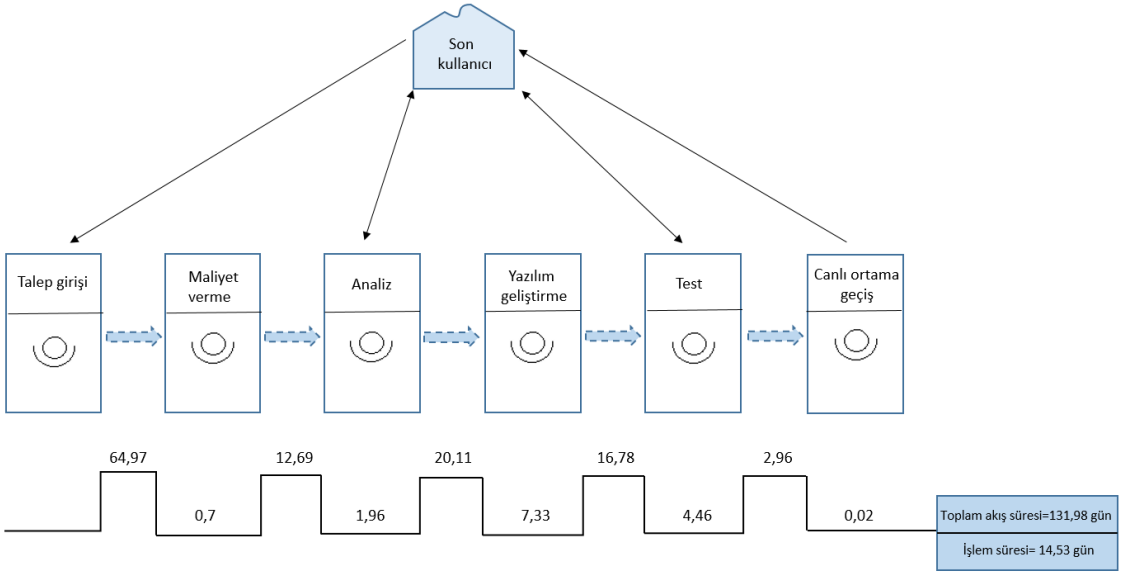
Şekil 4.6 : 1. Küme için mevcut değer akış diyagramı



Şekil 4.7 : 2. Küme için mevcut değer akış diyagramı



Şekil 4.8 : 3. Küme için mevcut değer akış diyagramı



Şekil 4.9 : 4. Küme için mevcut değer akış diyagramı

Mevcut durum değer akış diyagramları göz önüne alındığında en fazla zaman kayıplarının beklemeden kaynaklı olduğu gözlenmektedir. Bu durumda beklemelerin sebeplerinin ne olduğu konusuna odaklanılması gerekmektedir.

- Kullanıcının talebi girmesi ile başlayan süreçte talebin maliyetlendirilmesi aşamasına gelene kadar uzun süreler talep havuzunda beklediği görülmektedir. Talepler birim onayı ile havuza düşmektedir. İncelendiğinde ise birbirleriyle benzerliği, sistem geliştirme talebi olmaması, kullanıcının öncelik belirtmemesi gibi sebepler ile havuzda uzun süreler beklemektedirler.
- Maliyetlendirme aşamasından sonra analistin analize başlaması için talep uzunca bir süre beklemektedir. Aslında analist analize başlar başlamaz kartın durumunu analizde olarak değiştirmemektedir. Aslında bu sürenin bir kısmı analiz hazırlanma süresine dahil olmalıdır. Beklememin ilk sebebi taleplerin önceliklendirilmemiş olmasıdır. Taleplerin kısa aralıklarla önceliklendiriliyor olması da tam tersi durumlara sebep olabilmektedir. Çünkü sık sık önceliklendirme yapıldığında da analistin elinde devam ettiği işleri bırakarak sürekli başka bir işe başlaması gerekebilir. Bu da talebin uzun süre analizde kalmasına sebebiyet verebilir.
- Talebin analizde beklemesinin diğer sebepleri ise, ihtiyaçların en başında iyi analiz edilmemesi, ek talepler, projenin ne amaçla yapıldığının göz ardı edilmesi, talebin içeriğinin değişmesi ve yanlış anlaşılmalardr.
- Analiz süreci içerisinde dökümanlar eklenmeden analiz süreci tamamlanamamaktadır.
- Ayrıca yazılımcı kaynak sayısının eksikliği hem bekleme süresini hem de yazılım geliştirme sürecini uzatmaktadır.
- Eğer yazılımcı ve analist iyi bir iletişim içerisinde olmaz ise sadece doküman aracılığı ile hareket edildiğinde test sürecinde birçok yanlış anlamadan kaynaklı hata meydana gelmektedir.
- Süreçte analistler hem analiz hem de test sürecinde yer almaktadır.
- Ayrıca test sürecinin en başında analistin manuel yapması gereken test ortamına taşıma işlemleri yer almaktadır
- Analist tarafından testleri tamamlanan projeler kullanıcı testi için uzunca süre beklemektedirler. Kullanıcı testleri yapılabilmesi için personel ayırılamıyor olması bunda en büyük sebeptedir.

4.4.İsrafların belirlenmesi ve iyileştirme önerileri

Yazılım geliştirme projelerinde en sık rastlanan israf hatalardan kaynaklanmaktadır. Bunun yanında ekstra süreçler, bekleme, ekstra özellikler, kısmet yapılmış iş, görev değişikliği ve hareketten kaynaklı olan israflar bulunmaktadır. Bu bölümde bu 7 israfa ve israfları ortadan kaldırabilmek için iyileştirme önerilerine yer verilmiştir.

4.4.1. Hatalar

- 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.
- Sistem geliştirme talebi olmayan, teknik sorunlara yönelik olarak açılmış taleplerin uzunca süreler havuzda beklemesi.

Analiz sürecinde en önemli unsur kullanıcıya neden sorusunun sorulmasıdır. Çünkü projeyi tamamlamak demek her zaman amaca ulaşmak demek değildir. Neden sorusunun sorulması ile 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 kısmı teknik sorunları içermekte yani aslında sistem geliştirme talebi içerisinde yer almaması gerekmektedir. Tüm bunları elediğimizde bekleme süresinin yarı yarıya azalacağını düşünebiliriz.

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.

4.4.2. Ekstra süreçler

- Gereksiz adımlar
- Dökümantasyon

Maliyetlendirme süreci incelediğinde toplam işlem süresi içerisinde en az süreye sahip süreçtir. Her küme için ortalama 1 günü bile bulmamaktadır. Ancak burada maliyet onaylandıktan sonra gelen maliyet belirlendi adımı tamamen gereksiz ekstra bir süreç olduğundan süreçten çıkarılabilir.

Analiz sürecini detaylı incelediğimizde 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 adımı çıkardığımızda birinci küme için ortalama 0,12 gün, ikinci küme için 0,08 gün, üçüncü küme için 0,19 gün ve dördüncü küme için 0,15 gün zaman kazanılmış olacak.

Analiz ve yazılım süreci içerisinde dökümanların tamamlanmadan sürecin ilerletilememesi durumunu önleyebilmek için süreç ilerletildikten sonra dökümanların eklenebilmesi gibi seçenekler sunulabilmelidir. Tüm bunlar gözönünde bulundurularak analize başlama süresi ve analizde kalma süreleri kısaltılabilir. Şekil 4.10, Şekil 4.11 ve Şekil 4.12'de her bir süreçte yer alan gereksiz adımlar işaretlenmiş ve gelecek durum akış diyagramı oluşturulurken bu yeni süreçler üzerinden hesaplama yapılmıştır.

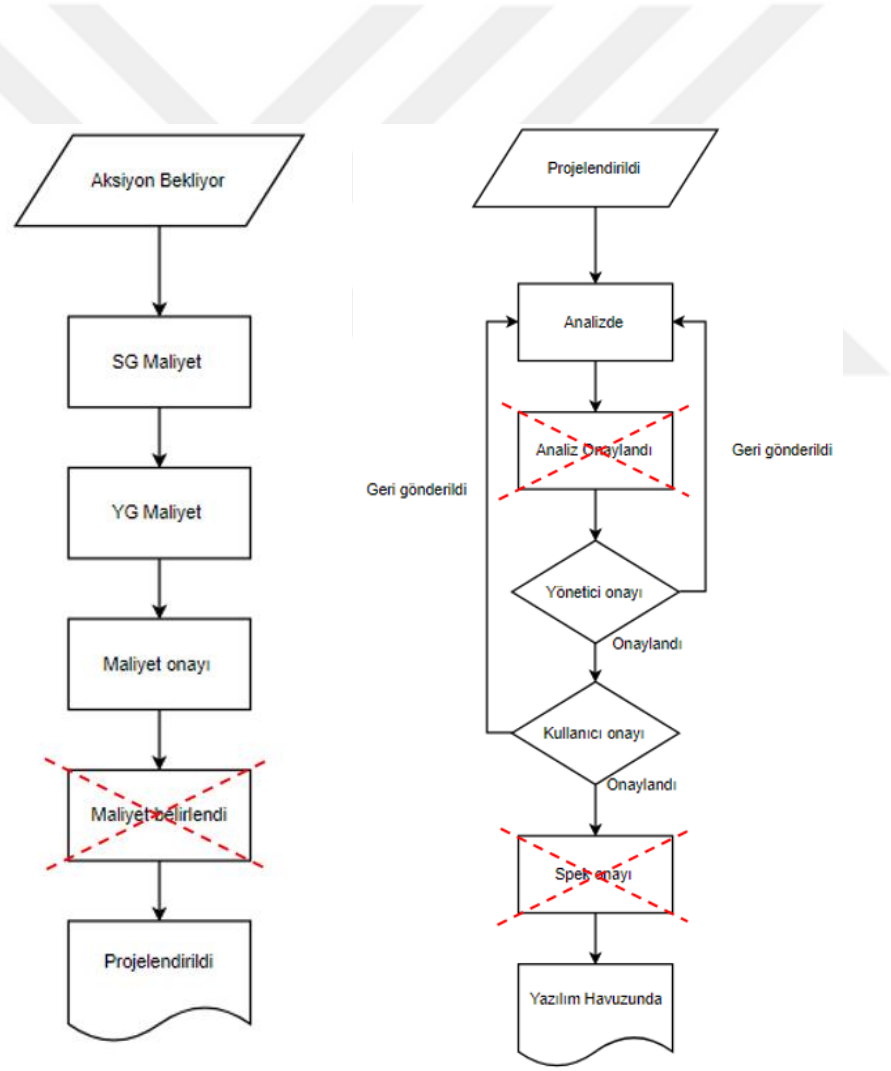
4.4.3. Bekleme

Kaynak sayısının arttırılması ile yazılıma başlama süresi ve yazılım geliştirme süreci yarı yarıya azaltılabilir.

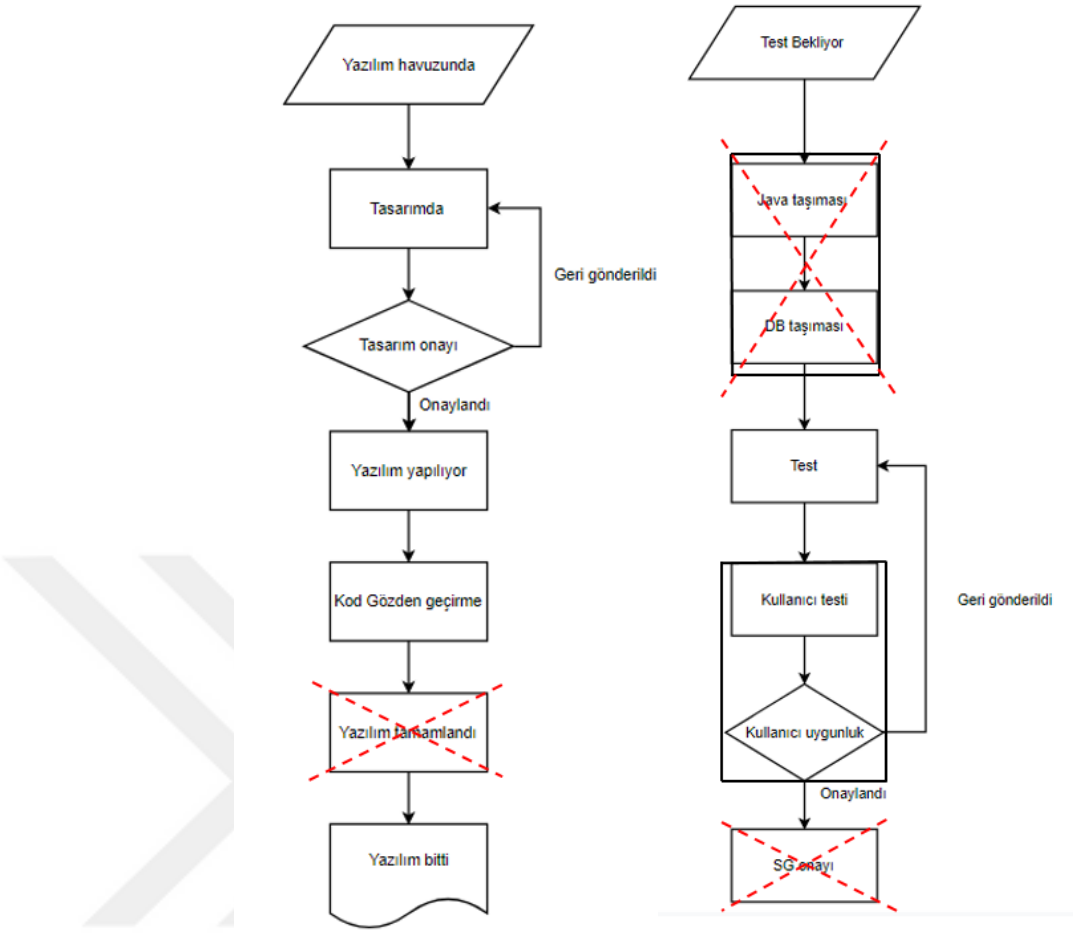
Süreçte analistler hem analiz hem de test sürecinde yer almaktadır. Test ekibinin kurulması ile analistlerin analize ve teste başlamak için bekledikleri süre yarı yarıya inecektir. Birden fazla projenin aynı anda yürütüldüğü göz önünde bulundurulduğunda yazılım ve test süreçleri de yarı yarıya azaltılabilecektir.

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

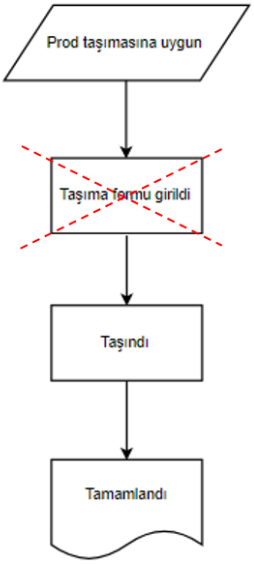
Kullanıcı testleri yapılabilmesi için personel ayırılamıyor olması taleplerin kullanıcı testi için beklemesinin en büyük sebebidir. Kullanıcı testleri sırasında iş birimi trafından o anda boş olan personel değil gerçekten bilen birini bulmak ve Bilgi Teknolojileri tarafında da işi anlatacak personeli sağlamak çok önemlidir. Testlere başlamadan önce BT personeli tarafından kullanılacak test dataları da belirlenmelidir. Kullanıcı test için bir personel ayrılması kullanıcı testine başlama süresini oldukça azaltacaktır. Test süreci içerisinde tespit edilen hatalar aslında en gözle görülebilir israf çeşididir.



Şekil 4.10 : Fazladan işlemlerin çıkarılması ile maliyet ve analiz süreci



Şekil 4.11 : Fazladan işlemlerin çıkarılması ile maliyet ve analiz süreci



Şekil 4.12 : Fazladan işlemlerin çıkarılması ile taşıma süreci

4.4.4. Ekstra özellikler

Yazılımcıların gönderilen analiz dökümanına uymayarak ekstra özellikler yazmaları da süreçte bir israfa sebep olmaktadır. Analiz dökümanında istenen özellikler dikkate alınarak fazladan kod yazılması istafın önüne geçecektir.

4.4.5. Kısmen yapılmış iş ve görev değişikliği

Önceliklendirmenin doğru zaman aralıkları ile yapılması kısmen yapılmış iş olarak adlandırılan israf çeşidine de bir örnek olarak sayılabilir. Çünkü analist ve yazılımcı elindeki işi kısmen tamamlamışken diğer işe geçiş yapar. Bunun yanında çalışanlar projelerini bırakarak başka bir projeye geçiş yaptıkları için görev değişikliği israf türü olarak ta sayılabilmektedir.

4.4.6. Hareket

Gün içinde yürütülen projelerin yanında canlı ortamda meydana gelen problemlerden kaynaklı olarak analist ve yazılımcılar harekete geçmek zorundadır. Bu da ellerinde yer alan projeyi bırakarak sorunlarla ilgilenmeleri anlamına geliyor. Projenin süresine problemin çözülme süresi de eklenmelidir. Hareket israfını ortadan kaldırabilmek için canlı ortamda meydana gelen problemler ile ilgilenmek üzere bir ekip kurularak, mevcutta yürütülen projelere etkisi ortadan kaldırılmış olacaktır.

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, gerçekler belirleninceye kadar bağlayıcı kararlar alınmaması 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.

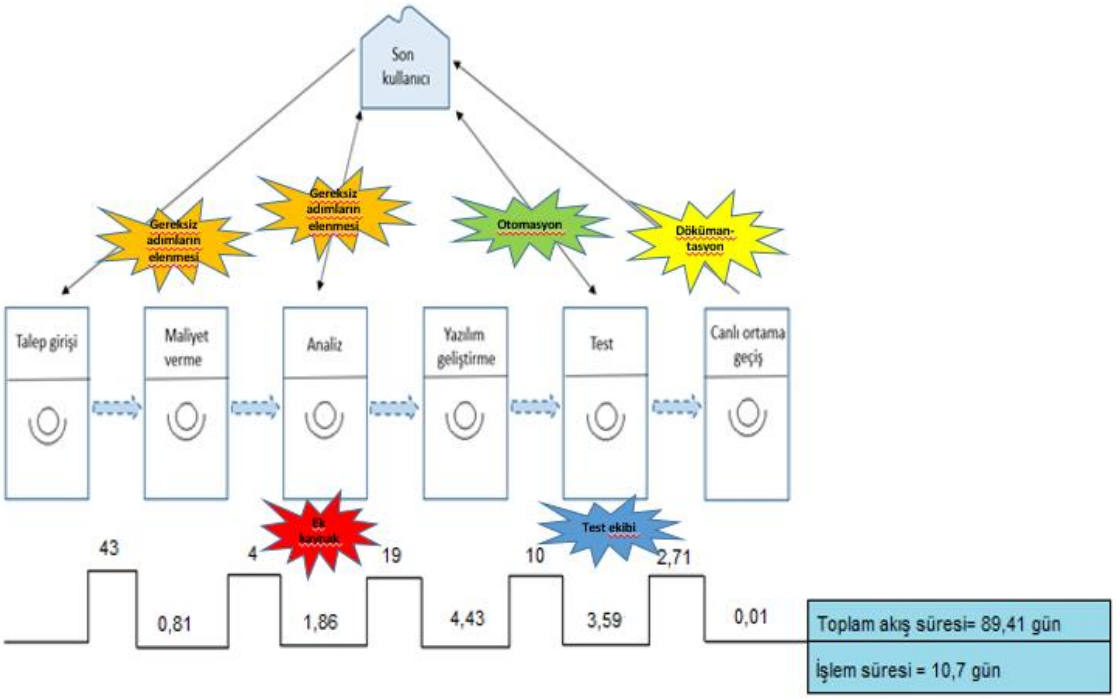
Çevrim süresini azaltmanın bir diğer ve daha maliyetli yolu ise kullanılan yazılım geliştirme metodunu değiştirmektedir. İncelediğimiz vakada şelale metodu bir diğer adıyla klasik yaklaşım kullanılmaktadır. Bu metodun en önemli özelliği bir faz tamamlanmadan bir diğer faza geçilememektedir. Bu durumda ara geçişlerde ya çalışan beklemektedir, ya da çalışanın iş yükü olduğu için proje beklemektedir. Ayrıca bir fazdan diğer faza dönüş yüksek maliyet gerektirir.

Alternatif bir yöntem olarak Çevik yazılım geliştirme metodu önerilebilir. Çevik metodu kısa sürede değerli çıktı üretmeye odaklanır. Projeler küçük parçalara ayrılarak birden fazla süreç paralel bir şekilde yürütülür. Bu durum hataların kısa sürede tespit edilerek çözüme kavuşturulmasında büyük rol oynar. Ayrıca klasik yönteme göre en büyük avantajı değişen gereksinim için oldukça esnek olmasıdır. Müşteriye sürekli geri bildirim verilmesi sebebi ile yanlış anlaşılmalarda büyük ölçüde azalır. Ancak yazılım geliştirme metodunu değiştirmek özellikle kurumsal yapıda firmalarda oldukça zordur. Yeni teknolojiler ve eğitimler gerektirdiğinden maliyetlidir ve çalışanların adaptasyon süreci de geçişi zorlaştıran etkenlerden biridir.

4.5. Gelecek Durum Değer Akış Haritalama

En uzun bekleme süresine sahip olan küme-1 için aşağıda yer alan iyileştirmeler doğrultusunda Şekil 4.13'teki gelecek durum değer haritası çizilmiştir. İyileştirmeler ile süreler kısalmış ve işlemler için gereken kaynak ihtiyacı en aza indirilmiştir.

- 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 0,17 gün kısaltılabilir.
- Süreçler içerisinde analizi hazırlayan analistin tekrar onay vermesi ve yöneticinin en son tekrar onay vermesi işlemlerinin süreçten çıkarılması ile küme-1 için analiz adımı 0,12 gün kısalacaktır.
- 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 küme-1 için bekleme süresinden 1,89 gün düşülebilecektir.
- 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.



Şekil 4.13 : Gelecek durum değer akış haritası

5. SONUÇ

Bu çalışmanın amacı, yazılım geliştirme projesinde mevcut değer akış haritasından yola çıkarak israfları belirleyebilmek ve bu israfları ortadan kaldırabilecek ve süreç iyileştirmeleri sağlayacak öneriler sunarak daha kısa çevrim süreli gelecek durum değer akış haritasını oluşturabilmektir. Bu amaçla bir metodoloji önerilmiştir. Mevcut ve gelecek durum haritalarının oluşturulabilmesi için öncelikle yazılım geliştirme süreçleri için değer tanımı yapılmıştır. Yazılım projeleri için değeri doğruluk, güvenilirlik, veri bütünlüğü ve güvenliği, teslim zamanı, minimum maliyet, kullanım kolaylığı ve geliştirilebilirliğe açık olması kriterleri ile tanımlanabilmektedir. Bu tanımlardan yola çıkarak her bir projenin maliyeti, tamamlanması için harcanan efor, projenin öncelikli olup olmadığı, amaç, son kullanıcıdan gelen ek talep sayısı, meydana gelen hataları düzeltmek için yapılan güncelleme sayısı, projede çalışan kaynak sayısı ve tecrübe yılları toplamından 8 farklı değişken tanımlanmıştır. Değişkenlerden yola çıkarak projeler gruplandırılmıştır. Toplamda 154 yazılım projesi için tanımlanan değişkenlerden yola çıkarak SPSS'te iki adımlı kümeleme analizi yapılmış ve sonucunda projeler 4 farklı kümeye ayrılmıştır. Projelere ayrılırken kullanılan değişkenlerin önem sırası Amaç, Öncelik, Maliyet, Efor, Kaynak, Ek talep sayısı, Güncelleme sayısı, Tecrübe yılı şeklinde olmuştur.

Dördüncü küme için en önemli değişken amaç iken birinci, ikinci ve üçüncü kümeler için en önemli değişken öncelik olmuştur. Ortaya çıkan 4 küme için mevcut değer akış haritaları çizilmiştir.

Mevcut durum değer akış haritalarının analizi israfların belirlenmesini ve bu israfları en aza indirilebilmek için ise iyileştirme önerileri sunulmasını sağlamıştır. Önerilerden yola çıkarak en fazla bekleme süresine sahip olan küme için süreç iyileştirmeleri yapılarak gelecek durum değer akış haritası çizilmiştir.

Genel olarak incelendiğinde israfı gereksiz kodlar, iyi tanımlanmamış ihtiyaçlar, bürokrasi, yavaş iç iletişim oluşturmaktadır. Müşteri ile sık sık iletişim, prototip oluşturma, kaynak sorununu anlama, gerçekler belirleninceye kadar bağlayıcı kararlar alınmaması 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.

Çevrim süresini azaltmanın bir diğer ve daha maliyetli yolu ise farklı bir yazılım geliştirme metodunu kullanmaktadır. Alternatif bir yöntem olarak Çevik yazılım geliştirme metodu önerilebilir. Çevik metodu kısa sürede değerli çıktı üretmeye odaklanır. Projeler küçük parçalara ayrılarak birden fazla süreç paralel bir şekilde yürütülür. Bu durum hataların kısa sürede tespit edilerek çözüme kavuşturulmasında büyük rol oynar. Ayrıca klasik yönteme göre en büyük avantajı değişen gereksinim için oldukça esnek olmasıdır. Müşteriye sürekli geri bildirim verilmesi sebebi ile yanlış anlaşılmalarda büyük ölçüde azalır. Ancak yazılım geliştirme metodunu değiştirmek özellikle kurumsal yapıda firmalarda oldukça zordur. Yeni teknolojiler ve eğitimler gerektirdiğinden maliyetlidir ve çalışanların adaptasyon süreci de geçişi zorlaştıran etkenlerden biridir.

Gelecek durum değeri akış haritası, yazılım işlem akışı, bilgi akışı ve ürün dönüşüm akışı hakkında net bir fikir vermiştir. Değer akışı haritalama işlemi verimli bir araçtır ve yalın bir işletmeye dönüşmenin ilk adımıdır ve bu yüzden de Bilgi Teknolojisi departmanında Yazılım geliştirme süreci için uygulanmıştır. Gelecekteki durum haritası için çevrim zamanına bağlı olarak, her adımda gereken tam zamanlar belirlenir ve tüm işlem akışındaki her adım arasındaki gereksiz sıra süreleri minimum seviyeye indirilir. Son kullanıcı ile yazılım geliştirme departmanı arasındaki bilgi ve iletişim akışı sıklığı artmakta, böylece yazılım ürün akışıyla ilgili süreçte daha fazla görünürlük elde edilmektedir.

Gelecekteki yeni durum haritasının faydaları, sürece dahil olan tüm kişilerin desteği ve katılımıyla ve şirkette yönetimin desteğiyle hissedilecektir. Yazılım geliştirme süreç akışında, değer akış haritalaması öncesinde ve gelecekteki durum değer akış haritası sonrasında önemli bir değişiklik oluşur ve bunun doğru bir şekilde uygulanması durumunda bunun faydaları şirketteki kullanıcılar tarafından hissedilecektir.

Projede yer alan değişkenlerin hem nümerik hem de kategorik olması iki adımlı kümeleme analizi yapılması konusunda bir kısıtlmaya sebep olmuştur. Farklı değer tanımlarından yola çıkarak değişkenler sadece nümerik veya sadece kategorik konuma

getirilerek farklı kümeleme analizi metodları kullanılabilir ve daha kısa çevrim süresine sahip yeni kümeler elde edilmesi sağlanabilir.

Yazılım geliştirme projeleri için değer odaklı iyileştirme çalışmaları oldukça kısıtlıdır, bu çalışmada hem iki adımlı kümeleme analizi yapılmış hem de değer odaklı iyileştirme yapılarak mevcut ve gelecek durum haritalarının oluşturularak yazılım geliştirme projeleri için değer odaklı iyileştirme alanında kaynak sağlanmıştır.

Bu araştırma, Bilgi Teknolojileri departmanı için yapılmıştır. Fakat insan kaynakları, finans, satış ve pazarlama, müşteri hizmetleri ve destek, vb. diğer alanlara da genişletilebilir. Maliyet ve benzeri performans ölçütlerinde önemli miktarda tasarrufla birlikte şirketin tamamındaki işleri yönetmede etkinlik sağlayacaktır. Zamanında ve daha iyi müşteri hizmeti ile kaliteli ürünler ortaya çıkarılabilecektir. Farklı bir gelecek önerisi olarak ise yazılım geliştirme metodolojisi olarak çevik metodu kullanan farklı bir şirkette süreç analizi ve kümeleme işlemleri yapılıp değer akış haritaları çizilebilir. Böylece klasik ve çevik yaklaşımlar kıyaslanarak iki metod arasındaki fark net bir şekilde ortaya konabilir.

KAYNAKLAR

Aktaş, E.U., Altunel, H., Elalmış, H.İ., Nişancı, S. & Yılmaz, C. (2018). Improvement of software issue record management process in an industrial context. *CEUR Workshop Proceedings, 220.*

Alahyari, H., Sivensson, R. & Gorschek, T. (2017). A study of value in agile software development organizations, *The Journal of Systems and Software,125*, 271-288.

Ali N, Petersen, K. & 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.

Ali N, Petersen, K. & Schneider, K. (2015). Evaluation of simulation-assisted value stream mapping for software product development: Two industrial cases, *Information and Software Technology*, *68*, 45-61.

Amazal, F., Idri, A. & Abran, A. (2014). An Analogy-based Approach to Estimation of Software Development Effort Using Categorical Data, *International Conference on SoftwareProcess and Product Measurement*, 2, (s.252-262). Morocco. doi: 10.1109/IWSM.Mensura.2014.31

Anand, G., Chandrashekar, A. & Narayanamurthy, G. (2014). Business process reengineering through lean thinking: A case study, *Journal of Enterprise Transformation, 4 (2),* 123-150.

Barki, H., Rivard, S., & Talbot, J. (1993). Toward an assessment of software development risk. *Journal of Management Information Systems*, 10(2), 203–225.

Bauch, C. (2004). Lean product development: Making waste transparent (Bitirme Tezi), Massachusetts Institute of Technology, Technical University of Munich.

Bekçi, İ. & Doğru, E. (2011). Değer yaratan faaliyetler açısından işletme başarısı: çimento sanayii işletmelerinde bir araştırma, *Journal of Süleyman Demirel University Institute of Social Sciences,13*, 169-192.

Boehm, B., (1988) "A Spiral Model for Software Development and Enhancement," Computer , vol. 21, no. 5, May 1988, pp. 61–72.

CMMI (2010).CMMI for Development, Version 1.3 (CMU/SEI-2010-TR-033). Retrieved February 20, 2020, from the Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661

Dadashzadeh, M. & Wharton T. (2012). A Value Stream Approach For Greening The IT Department, *International Journal of Management & Information Systems* (2nd ed., Vol. 16, pp.125-136). Oakland, USA.

Dingsoyr, D. & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery, *Information & Software Technology, 77,* 56-60.

Ebert, C. & Dumke, R. (2007). Software measurement: Establish - Extract - Evaluate - Execute, *Information & Software Technology, 77,* 56-60.

Haksever, C. & Chaganti, R. (2004). A Model of Value Creation: Strategic View, *Journal of Business Ethics, 49,* 291-305.

Hannan, M. (2011). Analysis of the collaborative activities in software development processes from the perspective of chronotopes, *Computers in Human Behavior,27,* 248-267.

Herdiyanti, A., Listyawati, N. R., & Astuti, H. M. (2017). Understanding the Total Value of Information Technology Services from the Perspective of Students and Academic Staffs. Procedia Computer Science, 124, 429-436.

Huang, S. & Han, W. (2008). Exploring the relationship between software project duration and risk exposure: A cluster analysis, *Information & Management, 45,* 175-182.

IBM SPSS. (n.d.). The SPSS TwoStep cluster component. White Paper, Retrieved January 20, 2020, from https://www.spss.ch/upload/1122644952_The%20SPSS%20TwoStep%20Cluster%20Component.pdf

ISO/IEC 33001:2015. (2015). Information technology — Process assessment — Concepts and terminology the International Organization for Standardization (ISO) and the International Electrotechnical Commission(IEC), https://www.iso.org/standard/37458.html

Jon, G., Chan, C. & Teng, C. (2012). The effect of clusters on the development of the software industry in Dalian, China, *Technology in Society, 34,* 163-173.

Kartal, E. (2015). *Sınıflandırmaya dayalı makine öğrenmesi teknikleri ve kardiyolojik risk değerlendirmesine ilişkin bir uygulama.* (Doktora tezi). İstanbul Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

Khan, P. M. & 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.

Khan, S. & Ahmad, A. (2017), Survey of State-of-the-Art Mixed Data Clustering Algorithms *IEEE TRANSACTIONS and JOURNALS.*

Khurum, M., Petersen, K., & Gorschek, T. (2014). Extending value stream mapping through waste definition beyond customer perspective. Journal of Software: Evolution and Process, 26(12), 1074-1105.

Mujtaba, S., Feldt, R., & Petersen, K. (2010). Waste and lead time reduction in a software product customization process with value stream maps. In

2010 21st Australian software engineering conference (pp. 139-148). IEEE.

Murphy, G.C. & Kersten, M. (2020). Towards Bridging the Value Gap in DevOps. *Lecture Notes in Computer Science, 12055,* 181-190.

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. & McCaffery, F. (2018). A hybrid assessment approach for medical device software development companies, *Journal of Software: Evolution and Process, 30 (7),* e.1219.

Özevre, M. (2008). İşletmelerde Değer Yönetimi, Beta Yayınları.

Papadopoulos, G. (2015). Moving from traditional to agile software development methodologies also on large, distributed projects, *Procedia - Social and Behavioral Sciences,175,* 455-463.

Patterson, R., F. (1989). *New Webster's Dictionary*: 1989 Edition Paperback. P.S.I. & Associates, Inc.

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

Poppendieck, M., Poppendieck, T.: (2006) Implementing Lean Software Development: From Concept to Cash. Addison-Wesley Professional, Upper Saddle River

Porter, M. E. (1985). The Competitive Advantage: Creating and Sustaining Superior Performance. NY: Free Press

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

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

SANGEM. (2012). *Yalın Yazılım Geliştirme*. Sanayi Geliştirme Merkezi (SANGEM), Sunum notları Retrieved from http://www.esinkap.net/5projepazari/sunumlar/salon4/1.oturum/ArGePP5_GB.pps

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

Schmidt, F. L., & Hunter, J. E. (1998). The validity and utility of selection methods in personnel psychology: Practical and theoretical implications of 85 years of research findings. *Psychological Bulletin*, 124(2), 262–274

Staron, M., Meding, W. & Caiman, M. (2013). Improving completeness of measurement systems for monitoring software development workflows. *Lecture Notes in Business Information Processing*, 133, 230-243.

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

Ülgen, H. & Mirze, K. (2007). *İşletmelerde Stratejik Yönetim*. İstanbu, Türkiye.

Vijayasarathy, L., & Butler, C. (2016). Choice of Software Development Methodologies. IEEE Computer Society - IEEE Software, September/October 2016, 86-94.

Walker, H. R. & Jhonson, W. L.& Leonard, S. (2006). Re-thinking the conceptualization of customer value and service quality within the service-profit chain, *Managing Service Quality,16*, 23-36. doi: 10.1108/09604520610639946

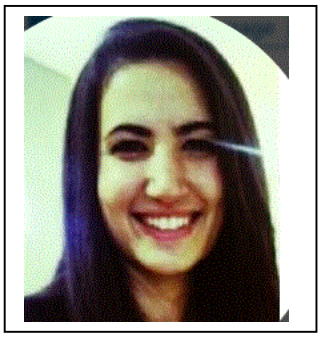
Walker, R., Johson, L. & Leonard, S. (2011). Customer-Driven Value Co-creation in Service Networks, *The Science of Service Systems*, *49*, 189-206. doi: 10.1007/978-1-4419-8270-4_11

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

Yadav, R., Mittal, M. L., & Jain, R. (2019). Lean practices in software development projects: A literature review. In AIP Conference Proceedings (Vol. 2148, No. 1, p. 030044). AIP Publishing LLC.

Zeithaml, V. A. (1988). Consumer Perceptions of Price, Quality, and Value: A Means-End Model and Synthesis of Evidence. Journal of Marketing, 52(3), 2–22. https://doi.org/10.1177/002224298805200302

ÖZGEÇMİŞ



Ad-Soyad **:** Ebru Ertek

Doğum Tarihi ve Yeri **:** 19.02.1992 - Şanlıurfa

E-posta **:** ebru92cuhadaroglu@gmail.com

ÖĞRENİM DURUMU:

- **Lisans** **:** 2016, İstanbul Arel Üniversitesi, Mühendislik-Mimarlık Fakültesi, Endüstri Mühendisliği
  : 2017, İstanbul Arel Üniversitesi, Mühendislik-Mimarlık Fakültesi, Makine Mühendisliği

MESLEKİ DENEYİM VE ÖDÜLLER:

- 01.2018-02.2020 tarihleri arasında Anadolubank şirketinde Sistem Geliştirme departmanında Sistem Analisti olarak çalışmıştır.
- 03.2020 tarihi itibariyle Etiya şirketinde Analist olarak çalışmaya başlamıştır.