T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

ÖN-UÇ YAZILIMLARININ ANALİZİ İÇİN YENİ BİR BELİRTİM MODELİ ÖNERİSİ

Kadir ÇAMOĞLU

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Tez Danışmanı: Dr. Öğr. Üyesi Rembiye KANDEMİR

EDİRNE-2020

KADİR ÇAMOĞLU'nun hazırladığı "ÖN-UÇ YAZILIMLARININ ANALİZİ İÇİN YENİ BİR BELİRTİM MODELİ ÖNERİSİ" başlıklı bu tez, tarafımızca okunmuş, kapsam ve niteliği açısından Bilgisayar Mühendisliği Anabilim Dalında bir Doktora tezi olarak kabul edilmiştir.

Jüri Üyeleri (Ünvan, Ad, Soyad):	İmza
Doç.Dr.Hilmi KUŞÇU	.....
Doç.Dr.Erdinç UZUN	.....
Dr.Öğr.Üyesi Rembiye KANDEMİR	.....
Dr.Öğr.Üyesi Aydın CARUS	.....
Dr.Öğr.Üyesi Fatih YÜCALAR	.....

Tez Savunma Tarihi: 27/12/2019

Bu tezin Doktora tezi olarak gerekli şartları sağladığını onaylarım.

İmza

Dr. Öğr. Üyesi Rembiye KANDEMİR
Tez Danışmanı .....

Trakya Üniversitesi Fen Bilimleri Enstitüsü onayı

Prof.Dr.Murat YURTCAN
Fen Bilimleri Enstitüsü Müdürü

T.Ü. FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ DOKTORA PROGRAMI
DOĞRULUK BEYANI

Trakya Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında, tüm verilerin bilimsel ve akademik kurallar çerçevesinde elde edildiğini, kullanılan verilerde tahrifat yapılmadığını, tezin akademik ve etik kurallara uygun olarak yazıldığını, kullanılan tüm literatür bilgilerinin bilimsel normlara uygun bir şekilde kaynak gösterilerek ilgili tezde yer aldığını ve bu tezin tamamı ya da herhangi bir bölümünün daha önceden Trakya Üniversitesi ya da farklı bir üniversitede tez çalışması olarak sunulmadığını beyan ederim.

13/01/2020

Kadir ÇAMOĞLU

Doktora Tezi
Kadir ÇAMOĞLU
T.Ü. Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

ÖZET

Bilgi Teknolojilerinde bir yazılım projesini başarıyla tamamlamak ve doğru ürünü ortaya koyabilmek ancak etkili bir gereksinim analiziyle mümkün olabilmektedir. Günümüzde kullanılan mevcut yazılım gereksinimi analiz ve belirtim yöntemleri sınırlı sayıdadır ve analizi uçtan uca kapsayarak analistlere yol gösterecek bir bütünlük içermemektedirler. Bu çalışmanın temel motivasyonu, özellikle son kullanıcı arayüzlerine dayalı iş otomasyon uygulamaları için yazılım projelerinde daha etkili gereksinim analiz yapılabilmesini sağlayacak bir gereksinim analizi metodu ortaya koymaktır.

Çalışmada öncelikle mevcut modellerin güçlü ve zayıf yönleri incelenmiş, gereksinim mühendisliği üzerine yazılmış olan standartlar ve bilgi birikimleri ele alınarak ideal bir metodolojide olması gereken özellikler belirlenmiştir. Belirlenen bu özellikler esas alınarak yeni bir metot önerisi kavramsal olarak tasarlanarak sahada iki örnek projede uygulanarak sınanmıştır. Ardından alınan geri bildirimler sonrasında metot iyileştirilerek daha yalın ve uygulanabilir şekilde yeniden tanımlanmıştır.

Çalışmanın başında belirlenen iyi bir analiz yönteminin sahip olması gereken 20 gereksinimin 17'si büyük oranda karşılanmıştır. Çalışmanın ilerleyen aşamalarında bu belirlenen gereksinimlere ek olarak ortaya çıkan bazı gereklilikler de göz önünde bulundurularak bir analiz belirtim modeli, ön ve detay analiz süreci, analize dayalı bir emek tahminleme yaklaşımı, analizin doğrulanmasına yönelik bir yaklaşım ve tüm bunların gerçekleştirilebilmesini sağlayacak bir yazılım ortaya koyulmuştur. Yeni

belirtim modelinin gerçekleştirilmesini sağlayan yazılım, gereksinim analizini uçtan-uca ele alması, analiste kılavuzluk etmesi, analiz öğelerini bütüncül bir bakışla doğrulaması ve otomatik olarak doküman üretebilmesi gibi özellikleriyle birlikte analizin başarımına önemli ölçüde katkıda bulunmaktadır.

Tüm bu çalışma sonuçlarına ve bulgulara dayanarak, mevcut modellerle karşılaştırmaların sonucuna bakıldığında önerilen modelin yazılım projelerinde uygulanmasının proje başarısı ve ürün kalitesine önemli oranda katkıda bulunacağı düşünülmektedir.

Yıl : 2020
Sayfa Sayısı : 184
Anahtar Kelimeler : yazılım gereksinim analizi, belirtim modeli, analiz süreçleri, gereksinim mühendisliği

Doctoral Thesis
Kadir ÇAMOĞLU
Trakya University Institute of Natural Sciences
Department of Computer Engineering

ABSTRACT

In Information Technology, successful completion of a software project and producing the right product is only possible with effective requirement analysis. The current software requirement analysis and specification methods used today are limited and do not include an end-to-end analysis to provide analysts with integrity. The main motivation of this study is to provide a requirement analysis method that will enable more effective requirements analysis in software projects, especially for front-end business automation applications.

In this study, first of all, the strengths and weaknesses of the existing models are examined, and the standards and knowledge written on requirements engineering are discussed and the features that should be in an ideal methodology are determined. Then, a new method proposal was conceptually designed and tested in the field with sample project applications and the method was improved after the feedback received.

We determined 20 requirements needed for a good analysis method at the start of the study, and found that 17 of these would be met to a large extent in this study. In the later stages of the study, in addition to these requirements, an analysis specification model, preliminary and detailed analysis process, an analysis based effort estimation system, an approach to validation of the analysis and a software that will enable all of these to be realized were introduced. The software, which enables the realization of the new requirement specification model, contributes significantly to the performance of the analysis with its features such as end-to-end analysis of requirements, guiding the analyst,

verifying the analysis elements with a holistic view and producing documents automatically.

Based on the results and findings of this study, it is thought that the application of the proposed model in software projects will significantly contribute to project success and product quality.

Year : 2020
Number of Pages : 184
Keywords : software requirements analysis, specification models, analyze processes, requirements engineering

ÖNSÖZ

Bu çalışmanın amacı 25 yıldan uzun bir süredir içinde bulunduğum Bilgi Teknolojileri alanında daha başarılı projeler ve daha iyi ürünler geliştirilmesine katkıda bulunması ümidiyle, yeni bir yazılım gereksinim belirtim yönteminin ortaya koyulmasıdır.

Bu çalışmanın gerçekleştirilmesinde büyük emeği olan, 2010 yılında Trakya Üniversitesinde başladığım doktora eğitimimden bugüne kadar her konuda beni yönlendiren ve desteğini esirgemeyen çok değerli Danışman Hocam Dr. Öğr. Üyesi Rembiye KANDEMİR'e en derin minnet ve teşekkürlerimi sunmayı bir borç bilirim.

Ayrıca çalışma süresince hem değerlendirme hem de yönlendirmeleriyle katkıda bulunan jüri üyeleri Doç. Dr. Hilmi Kuşçu ve Dr. Öğr. Üyesi Aydın Carus hocalarıma, çalışmada yer alan örnek uygulamaların proje üyelerine hem doktora çalışmasına hem de örnek uygulamaların yürütülmesine izin ve destek veren, ikinci ailem olarak gördüğüm Bilge Adam Bilişim Teknolojileri Grubunun yöneticilerine ve çalışma arkadaşlarıma teşekkürlerimi sunarım.

Ve son olarak çalışmamda desteğini ve bana olan güvenini benden esirgemeyen bu hayattaki en büyük şansım olan aileme sonsuz teşekkürler.

Kadir ÇAMOĞLU

İstanbul, Aralık 2019

İÇİNDEKİLER

SİMGELER ve KISALTMALAR DİZİNİ

Kısaltmalar

BABOK ®	A Guide to the Business Analysis Body of Knowledge ® – İş Analizi Bilgi Birimi İçin Bir Kılavuz
BEBK	Proje Ekibi Başarım Katsayısı
CRM	Customer Relationship Management – Müşteri İlişkileri Yönetimi
CRUD	(Create – Retrieve – Update –Delete)
EP	Entegrasyon Puanı
ERP	Enterprise Resource Planning – Kurumsal Kaynak Planlama
FAK	Faktör Ağırlık Katsayısı
FKBP	Faktör Karmaşıklık Baz Puanı
FKHP	Faktör Karmaşıklık Ham Puanı
FKP	Faktör Karmaşıklık Puanı
HOS	Higher Order Software – Yüksek Öncelikli Yazılım
IEEE	Institute of Electrical and Electronics Engineers
İK	İnsan Kaynakları
İOG	İşlevsel olmayan gereksinim
İOGK	İşlevsel Olmayan Gereksinim Katsayısı
İÖB	İşlevsel öz birim
İÖBKHP	İÖB Karmaşıklık Ham Puanı
İÖBNKP	İÖB Net Karmaşıklık Puanı

KPAGDK	Karmaşıklık Puanı Adam Gün Dönüşüm Katsayısı
PSL	Process Specification Language – Süreç Belirtim Dili
PTKEP	Proje Toplam Karmaşıklık Emek Puanı
PTKP	Proje Toplam Karmaşıklık Puanı
RSL	Requirements Specification Language – Gereksinim Belirtim Dili
SADT	Structured Analysis and Design Technique – Yapısal Analiz ve Tasarım Tekniği
SAMM	Scenario and Model Management – Senaryo ve Model Yönetimi
SDLC	Software Developmennt Life-Cycle – Yazılım Geliştirme Yaşam Döngüsü
SLA	Service Level Agreement – Servis Seviyesi Anlaşması
SRS	Software Requirements Specifications, Yazılım Gereksinimleri Teknik Özellikleri
SWEBOK	Software Engineering Body of Knowledge – Yazılım Mühendisliği Bilgi Birikimi
UML	Unified Modeling Language – Birleşik Modelleme Dili

ÇİZELGELER DİZİNİ

ŞEKİLLER DİZİNİ

BÖLÜM 1

GİRİŞ ve AMAÇ

Günümüzde karmaşık iş süreçlerini yürütmek için kullanılan uygulamalar ağırlıklı olarak web uygulaması geliştirilmektedir. Bu uygulamalar çoğunlukla kullanıcı etkileşimli, nesneye dayalı programlama temeline dayanan ve çok katmanlı mimariyle geliştirilmiş uygulamalar olarak karşımıza çıkmaktadırlar. Bahsi geçen uygulamalarda yazılım geliştirme süreci, yazılım geliştirme yaşam döngüsü (Software Development Life Cycle - SDLC) adı verilen gereksinim analizi, mimari tasarım, geliştirme, test, entegrasyon ve bakım adımlarıyla gerçekleştirilmektedir (Leffingwell & Widrig, 2003).

Yazılım geliştirme sürecinin ilk adımı olarak tanımlanmış olan gereksinim analizi, başarılı bir yazılım projesi gerçekleştirmek ve doğru ürünü ortaya koyabilmek için önem verilmesi gereken en kritik aşama olarak karşımıza çıkmaktadır (The Standish Group).

Gerçek zamanlı, çevrim-içi, görev-kritik, oyun, gömülü sistemler gibi birçok farklı geliştirme türü için hem yazılım geliştirme süreçleri hem de buna bağlı olarak çalışacak olan gereksinim analizi metodolojisi değişiklik göstermektedir. Bununla beraber günümüzde hem literatür çalışmalarında hem de sanayideki uygulamalarda geleneksel yöntem ve çevik yöntem olmak üzere iki temel yaklaşım üzerinde yoğunlaşılmaktadır.

Günümüzdeki yazılım mimarilerinde veri yapısı, iş mantığı ve kullanıcı arayüzü mümkün olduğunca birbirinden soyutlaştırılmaya çalışılmakta, böylece geliştirme, bakım ve iyileştirme çalışmaları paralel ve birbirinden bağımsız olarak yürütülebilmektedir. Ön-uç ve arka-uç (front-end and back-end) olarak nitelendirilen bu yaklaşım veri işlemlerini ve karmaşık iş kurallarının çalıştırılmasını arka-uç denilen katmana bırakmakta,

kullanıcıyla etkileşimi ve kullanıcı veri giriş kontrolünü de ön-uç katmanıyla çözmektedir.

Literatürde yapılan araştırmalar yazılım gereksinimlerinin belirtimine yönelik olarak en çok kullanılan üç yöntemin kullanıcı hikayeleri (user story), kullanım vakaları (use case) ve Institute of Electrical and Electronics Engineers (IEEE) kurumunun 830-1998 standardıyla tanımlanan düz metin yaklaşımıdır (IEEE-STD 830-1998; Bittner & Spence, 2002).

Bu çalışmada, özellikle ön-uç uygulamalarda kullanılması amaçlanan ve yukarıdaki her üç gereksinim belirtim modelinin güçlü yönlerini esas alan yeni bir yazılım gereksinim belirtim modeli önerilmektedir.

1.1. Yazılım Geliştirme Süreç Modelleri

1970 yılında Dr. Winston Royce, yazılım geliştirme sürecinin iyileştirilmesi üzerine şelale modeli (Waterfall Model) adında bir yaklaşım ortaya koymuştur. Bu yaklaşım yazılım geliştirme sürecini birbiriyle ilişkili birtakım aşamalara bölmekte ve bu aşamaların belirli bir sırayla yapılması gerektiğini söylemektedir (Leffingwell & Widrig, 1999; Leffingwell & Widrig, 2003).

Ortaya konulan bu modelde yazılım geliştirme için bahsedilen süreçler sırasıyla şunlardır:

1. Gereksinimlerin edinimi,
2. Tasarımın geliştirilmesi,
3. Kodlama ve birim testi,
4. Sistem entegrasyonu,
5. Operasyon ve bakım.

1988'de Barry Boehm'in yaptığı bir çalışmada ise risk tabanlı bir yaklaşımla, şelale modelinde belirtilen fazlar artırımlı bir şekilde çalıştırılarak ara prototipler ortaya koymaya dayanır ve adına da Spiral model denmiştir. Spiral modelin şelale modelinden

temel farkı, ortaya konacak ürünle ilgili işlerin belirli fazlara bölünmesi ve bu fazlarda nihai ürünle ilgili ara çıktılar ortaya konması ve her fazda bir önceki fazda ortaya konan prototipin üzerine yeni özellikler eklenmesi şeklindedir (Boehm, 1998).

Daha sonraları Kruchten (1995) yinelemeli (iterative) yaklaşımı ortaya koymuş ve SDLC aşamalarına ayırıp, bu aşamaların birbirleriyle ilişkili bir biçimde gerektiği kadar tekrarına dayalı bir modeli ortaya koymuştur. Kruchten'in ortaya koyduğu bu aşamalar şöyledir:

1. Başlama (Inception)
2. Detaylandırma (Elaboration)
3. Yapım (Construction)
4. Geçiş (Transition)

2001 yılında aralarında Kent Beck, Mike Beedle, Ken Schwaber, Martin Fowler'ın da bulunduğu yazılım sektörünün öncülerinden 17 isim Utah'da bir araya gelerek bir çalışma yapmış ve bu çalışmanın sonunda Çevik Manifesto'yu yayınlamışlardı. Bu manifestoyla diğer yazılım geliştirme metodolojilerine alternatif olarak yeni bir yaklaşım ortaya koymuşlardır. Bu yaklaşımda kısa süreli yinelemeler (iteration), yineleme sonunda müşteriye çalışır bir ürünün teslimatı, müşteri için değer üretmeye odaklılık ve yakın çalışma gibi kavramlar esas unsurlar olarak ortaya konulmuştur (http://www.agilemanifesto.org). Çevik (agile) adı verilen bu yeni yaklaşımda Scrum, Kanban, Uç Programlama (Extreme Programming) başı çeken yöntemlerdendir (Hammarberg & Sundén, 2014; Pichler, 2010; Wallace, Raggett & Joel Aufgang, 2002).

The Standish Group'un "Chaos Manifesto 2011" isimli çalışmasında çevik ve geleneksel yöntemde proje başarıları incelenmiştir. Zamanında, kapsamında ve bütçesinde belirlenen kalitede biten projelerin geleneksel yöntemde %14 iken çevik yaklaşımlarda bu oranın %42 olduğu tespit edilmiştir. Diğer taraftan Versionone firması tarafından desteklenen başka bir araştırmada ise çevik süreçlerin kullanımının 2013 yılından 2014 yılına %19'dan %24'e yükseldiği gözlemlenmiştir.

Bu iki temel yaklaşımın dışında, sektörde ve literatürde bu yaklaşımların belirli uygulamalarını alarak ayrıca bir süreç oluşturmak da mümkündür (Çamoğlu, Akbayır, Yücalar & Bayraklı, 2010).

Bu araştırmalar ve bilgiler ışığında çalışmanın konusu ağırlıklı olarak çevik süreçler ve çevik süreçlerde kullanılan gereksinim modelleme yaklaşımları olan kullanıcı hikayeleri ve kullanım senaryoları olarak belirlenmiştir.

1.2. Yazılım Gereksinim Analizi

Yazılım Mühendisliği alanında en temel bilgi birikimi olarak kabul edilen Yazılım Mühendisliği Bilgi Birikimi (Guide to the Software Engineering Body of Knowledge - SWEBOK) Kılavuzunun üçüncü sürümünde 15 bilgi alanı (Konwledga Area) bulunmaktadır ve bu bilgi alanlarının ilki "Yazılım Gereksinimleri (Software Requirements) olarak geçmektedir (Bourque, Fairlay, 2014).

Uluslararası iş analizi enstitüsü (IIBA® –The International Institute of Business Analysis™) iş analizi bilgi birikiminde (BABOK® –A Guide to the Business Analysis Body of Knowledge®) iş analizinin tanımını şu şekilde yapmaktadır: "İş analizi, ihtiyaçları tanımlayıp onlara çözüm önererek paydaşlara değer katacak ürünler ortaya koymak üzere yapılan çalışmalardır (BABOK, 2015)."

Bu tezin kapsamı içerisindeki en kritik bileşenlerden biri de "gereksinim"dir. Gereksinim tanımı IEEE'nin 610.12-1990 nolu standardında aşağıdaki şekilde tanımlamaktadır (IEEE-STD 610.12-1990):

1. Bir problemi çözmek ya da bir amaca ulaşmak için kullanıcı tarafından ihtiyaç duyulan durum (condition) ya da kabiliyet (capability),
2. Bir kontrat, şartname ya da diğer resmi nedenler dolayısıyla bir sistem ya da sistem bileşeni tarafından yerine getirilmesi zorunlu olan durum ya da kabiliyet,
3. 1. ve 2. maddelerdeki durum ya da kabiliyetin yazılı olarak ifade edilmesi.

Çalışmada literatürdeki tüm bu tanımlar esas alınarak önerilecek modelin en doğru şekilde kurgulanmasına çalışılmıştır. Bu tezin 2. bölümünde gereksinim tanımına, gereksinim mühendisliğine ve gereksinim analizine ayrıntılı olarak değinilmiştir.

1.3. Proje Tipleri ve Yazılım Proje Mimarileri

Alexander'a göre her türlü yazılım ve proje tipi için geçerli olabilecek tek bir gereksinim toplama yöntemi yoktur. Farklı uygulama ve proje tiplerinin o tipe uygun yöntemlerle analiz yapılması daha uygun olacaktır. Alexander uygulama geliştirmek üzere yapılabilecek proje tiplerini kendi yazılımını geliştirme (in-house software development), ürün ya da servis geliştirme (product or service development), müşteriye özel yazılım geliştirme (custom development), paket program geliştirme (commercial off the shelf), alt üstlenici olarak geliştirme (subcontract) ve bakım (maintenance) olarak gruplamıştır (Alexander, 2009).

Bir taraftan yazılım projeleri yukarıdaki gibi sınıflanırken diğer taraftan yazılım projeleriyle ortaya konacak olan uygulama tipleri de ayrıca kategorilere ayrılmaktadır. Yazılım temel olarak sistem yazılımları ve uygulama yazılımları olarak ikiye ayrılmaktadır. Sistem yazılımları grubuna en temelde işletim sistemleri, derleyici gibi yazılımlar girmektedir. Uygulama yazılımları ise kelime işlemci, veritabanı yönetim paketleri, ofis otomasyon yazılımları, yayıncılık yazılımları gibi yazılımlardan oluşmaktadır. Uygulama yazılımları da kendi içinde genel amaçlı uygulama yazılımları ve özel amaçlı uygulama yazılımları olarak ikiye ayrılmaktadır. Genel amaçlı yazılımlara kelime işlemci, elektronik tablolama, veritabanı ve dosya yönetim yazılımları, sunum yazılımları örnek verilebilir. Diğer taraftan özel amaçlı yazılımlara da masaüstü yayıncılık, grafik ve animasyon yazılımları örnek verilebilir. Özel amaçlı yazılımlara ayrıca iş uygulamaları yazılımları da dahil edilmektedir. Bu başlık altında da kurumsal kaynak planlama (Enterprise Resource Plannig - ERP), müşteri ilişkileri yönetimi (Customer Relationship Management - CRM) gibi yazılımlar bulunmaktadır (Arora & Bansal, 2005).

Tezin çıktıları tüm yazılım tiplerini ve uygulama sınıflarını belirli oranda destekleyebilecek olmakla beraber öncelikli hedef müşteriye özel geliştirilen iş uygulama yazılımları olarak belirlenmiştir.

Yazılım mühendisliğinde katmanlı mimari uygulamalarında ön-uç ve arka-uç yaklaşımının amacı sunum katmanıyla veri erişim katmanının birbirinden ayrılmasını sağlamaktır. Ön-uç, kullanıcıyla uygulama arasındaki arayüz olarak çalışır. Çok katmanlı mimaride kullanıcıyla sistem arasında pek çok katman olması mümkündür. Kullanıcıya kullanıcı dostu bir arayüz sunarak katmanlardan bağımsız olarak yönetilebilecek arayüzler geliştirilebilmesi bu yaklaşımla mümkün olabilmektedir (Wikipedia, Front and Back Ends).

Tezde arka uçla ilgili de gereksinimler tamamen kapsam dışı bırakılmamakla birlikte temel olarak ön uç uygulamaların gereksinim analizi üzerinde durulmaktadır.

1.4. Yazılım Gereksinim Mühendisliğinin Yazılım Proje Başarımındaki Önemi

The Standish Group isimli bir araştırma şirketi 1990'lı yılların başından beri küçük, orta ve büyük ölçekli şirketlerde yine küçük, orta ve büyük ölçekli yazılım projelerini incelemekte ve yazılım projelerindeki başarılarla ilgili raporlar ve manifestolar yayınlamaktadır. Yayınlanan bu raporlar Amerika Birleşik Devletleri odaklı olmakla beraber genel bir fikir vermesi açısından önemlidir (The Standish Group Chaos Report, 2015). Grubun yayınladığı raporda 365 katılımcının ve 8380 projenin yer aldığı örnek grubuna göre, yazılım projelerinde gelinen son durumda zamanında, bütçesinde, kapsamında ve belirlenen kalite çerçevesinde tamamlanan projelerin oranı sadece %16,2 olarak belirtilmiştir. Projelerden %31,1'i iptal edilen projeler olarak, geri kalan %52,7'lik orandaki projelerin ise ya süresini ya bütçesini aşmış ya da sınırlı kapsamla devreye girmiş projeler olarak nitelendirilmiştir (The Standish Group Chaos Manifesto 2013).

Rapora göre başarısız projelerdeki önemli faktörlerin en başında kullanıcı girdilerinin eksikliği, tamamlanmamış gereksinimler, üst yönetim desteğinin yetersiz oluşu, gerçekçi olmayan beklentiler, açık olmayan hedefler gibi gereksinim ve paydaş yönetimine işaret eden başlıklar yer almaktadır (The Standish Group Chaos Manifesto 2013).

1.5. Tezin Kapsamı

Bu tezin amacı ön-uç iş uygulamaları yazılımları için, çevik yöntem temeline dayanan, yazılım gereksinim analizine yönelik, bir süreç ve belirtim modeli önermektir. Yazılım gereksinimlerinin proje başarısındaki payı göz önünde bulundurulduğunda, etkili bir analiz sürecinin olumlu katkı yapacağı düşünülmektedir.

Tezin kapsamı öncelikli olarak bu amaca hizmet edecek bir belirtim modeli ortaya koyulmasıdır. Ancak belirtim modeli tek başına eksik kalacağından, belirtim modelinde yer alması gereken çıktıları ürütmek için gerekli olan aktivitelerin ve sürecin de kapsama dahil edilmesine karar verilmiştir. Ayrıca yazılım projelerinde proje bütçesi ve zamanının doğru hesaplanmasına katkı verebilmek için, modele uygun bir emek tahminleme yaklaşımı da çalışmanın içine alınmıştır. Son olarak bu yeni modelle çalışma yapacak araştırmacı ve saha uygulamacılarının işlerini kolaylaştırmak adına modelle ilgili temel öğelerin geliştirilmesini, depolanmasını, doğrulanmasını ve raporlanmasını destekleyecek bir yazılımın geliştirilmesi de tezin kapsamına dahil edilmiştir.

Tez çalışmasının 2. bölümünde gereksinin analizi, gereksinim mühendisliği ve gereksinim kavramları üzerinde ayrıntılı olarak durulmakta ve ilişkili kavramlarla birlikte detaylı olarak açıklamalar yapılmaktadır.

Çalışmanın 3. bölümde mevcut gereksinim süreçleri ve gereksinim belirtim modelleri incelenmekte, modeller ve süreçler birbirleriyle karşılaştırılmakta ve artı ve eksi yönleriyle ortaya konulmaktadır.

Tezin 4. bölümünde ise çalışmanın konusu olan yeni gereksinim belirtimi modeli, gerekçesi, dayanakları, temel bileşenleri, örnek bir proje uygulamasıyla birlikte tüm ayrıntılarıyla tanımlanmaktadır.

Ardından 5. bölümde önerilen model ve iki örnek projede uygulaması üzerine sonuçlar ve tespitler ortaya koyulmaktadır.

Tezin 6. bölümünde bir önceki bölümde anlatılan örnek proje uygulamalarında edinilen geri bildirimler doğrultusunda modelde yapılan iyileştirmeler ve yeni modeli desteklemek üzere geliştirilen yazılımdan bahsedilmektedir.

Son bölümde ise çalışma ile ortaya koyulan model ana hatlarıyla tekrar ele alınmış, tespit edilen avantaj ve dezavantajlarıyla birlikte daha sonrasında yapılabilecek çalışmalarla ilgili görüşler sunulmuştur.

BÖLÜM 2

YAZILIM GEREKSİNİM ANALİZİNİN TEMEL ÖĞELERİ

Bu bölümde yazılım gereksinim analizinin ve temel bileşenlerinin tanımı, ilişkili kavramları, yazılım gereksinim analiz süreci ve ilişkili olduğu diğer yazılım geliştirme disiplinleriyle ilgili kapsamlı açıklama yapılmaktadır.

2.1. Yazılım Gereksinim Mühendisliği

Yazılım Mühendisliği alanında en temel bilgi birikimi olarak kabul edilen SWEBOK kılavuzunun üçüncü sürümünde 15 bilgi alanı (Knowledge Area) bulunmaktadır (Borque, Fairlay, 2014):

1. Yazılım gereksinimleri (software requirements),
2. Yazılım tasarımı (software design),
3. Yazılım yapımı (software construction),
4. Yazılım testi (software testing),
5. Yazılım bakımı (software maintenance),
6. Yazılım konfigürasyon yönetimi (software configuration management),
7. Yazılım mühendisliği yönetimi (software engineering management),
8. Yazılım mühendislik süreçleri (software engineering process),
9. Yazılım mühendisliği model ve metotları (software engineering models and methods),
10. Yazılım kalitesi (software quality),
11. Yazılım mühendisliği profesyonel uygulamaları (software engineering professional practice),

12. Yazılım mühendisliği ekonomisi (software engineering economics),
13. Hesaplamanın esasları (computing foundations),
14. Matematiksel temeller (mathematical foundations),
15. Mühendisliğin esasları (engineering foundations).

Yukarıdaki listede de görüleceği üzere "Yazılım Gereksinimleri" bir numaralı başlık olarak kılavuzdaki yerini almıştır.

Aurim ve Wohlin (2005), "Engineering and Managing Software Requirements" kitabında gereksinim mühendisliğinden şu şekilde bahsederler: "Gereksinim mühendisliği müşterinin problemini çözmek için ortaya doğru yazılımın konulabilmesi açısından yazılım tasarımı ve geliştirmenin en elzem aşamalarından biri olarak kabul edilir."

Buna ek olarak Young (2004), "The Requirements Engineering Handbook" isimli kitabında gereksinim mühendisliğini, süreç, rol, yöntem gibi boyutlarıyla detaylıca anlatmıştır.

Wahono (2003) gereksinim mühendisliği için, problemi analiz etmek, sonuçları çeşitli biçimlerde yazılı hale getirmek, kazanılan kavrayışın doğruluğunu kontrol etmekten oluşan sistematik bir süreç olarak tanımlamaktadır.

Benzer şekilde Bose, Kurhekar ve Ghoshal (2014) "Agile Methodology in Requiremnets Engineering" kitabında gereksinim mühendisliği hakkında müşterinin bir sistemden beklediklerini ya da o sistemle ilgili belirttiği kısıtları tanımlayan süreç olarak bahsetmekte ve gereksinim mühendisliği sürecinin ana amacının bilgi paylaşımı için sistem gereksinim dokümanı ortaya koymak olduğunu ifade etmektedir.

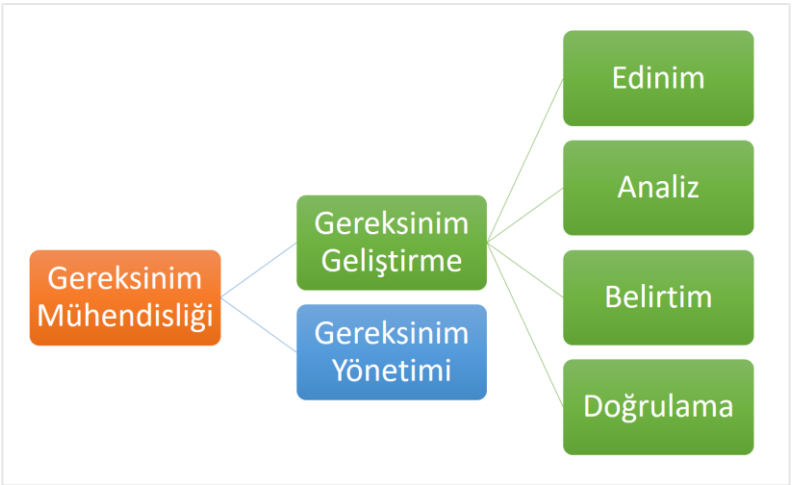
Gereksinim mühendisliği süreç, ürün ve projenin olduğu her yerde olan bir konu olmakla beraber halen tanımı, işleyişi ve bileşenleriyle ilgili araştırmalar devam etmektedir.

2.1.1. Gereksinim Mühendisliğinin Yapısı

SWEBOK yazılım gereksinimleri bilgi alanının alt başlıklarını şu şekilde tanımlamaktadır (Bourque and Fairley, 2014):

1. Yazılım Gereksinimlerinin Temelleri,
2. Gereksinim Süreci,
3. Gereksinimlerin Edinimi,
4. Gereksinimlerin Analizi,
5. Gereksinimlerin Belirtimi,
6. Gereksinimlerin Doğrulanması,
7. Uygulamaya İlişkin Konular.

Bilgi birikimi yazılım gereksinimleriyle ilgili temel tanımları Şekil 2.1.1.1. de görülebliecği gibi verdikten sonra gereksinim sürecini tarif etmektedir. Ardından gereksinimlerin edinimi başlığıyla devam eder ve gereksinim analizinin ardından gereksinim belirtimine değinir. Son olarak değindiği konu gereksinimlerin doğrulanmasıdır (Bourque and Fairley, 2014).



Şekil 2.1.1.1. SWEBOK Gereksinim Mühendisliği Yapısı (Bourque and Fairley, 2014)

Bilgi birikiminde tanımlar içerisinde ilk olarak ürün ve süreç gereksinimlerinden bahsedilmektedir. Kılavuzun tanımına göre geliştirilecek olan yazılımla ilgili parametreleri ortaya koyan gereksinimler ürün gereksinimleridir. Örneğin, "yazılım, öğrencilerin bir derse kayıt olmadan önce, öğrencinin o dersle ilgili tüm ön gereksinimleri yerine getirdiğini kontrol etmelidir."

Süreç gereksinimi ise yazılım geliştirme aşamasındaki kısıtları ifade etmektedir. Örneğin, "Yazılım Microsoft .NET teknolojisi kullanılarak geliştirilmelidir."

Kılavuzda daha sonra işlevsel/işlevsel olmayan gereksinim ayırımına gidilmektedir. SWEBOK işlevsel gereksinimleri yazılımın gerçekleştirmesi gereken işlevler olarak tanımlamaktadır. Örneğin bir metnin çözümlenmesi ya da iki sayının toplanması ya da başvuru bilgilerinin sisteme kaydedilmesi yazılımın işlevselliğiyle ilgilidir. İşlevsel olmayan gereksinimleriyse genellikle performans, bakım, güvenlik, sağlamlık gibi başlıklarla karşımıza çıkan karakteristik özellikler şeklinde özetlemektedir (Bourque and Fairley, 2014).

SWEBOK'da (2014), gereksinimler anlatılırken bahsedilen bir diğer konu da "Emergent Properties" olarak karşımıza çıkmaktadır. "Emergent Properties", yazılımın tek bir bileşeni tarafından yerine getirilemeyecek, birden fazla bileşenin bir araya gelerek ortaya koyacağı, tek başına tanımlanması mümkün olmayan özellikleri ifade etmektedir.

Kılavuzda yer alan bir diğer başlık, "ölçülebilir gereksinimler" olarak tanımlanmaktadır. Bu başlık altında gereksinimlerin mümkün olduğunca ölçülebilir ve test edilebilir şekilde tanımlanması gerektiğine dikkati çekmektedir. Kılavuzda belirsiz ve doğrulanamayacak gereksinimlerden kaçınılması gerektiği vurgulanmaktadır. Örneğin, "yazılım güvenilir olmalı", "yazılım kullanıcı dostu olmalı", vb. gibi. Nasıl kodlanacağı, geliştirileceği ve test edileceği belli olmayan bu gereksinimlerin aynı zamanda test edilemeyeceğine de dikkati çekmektedir.

Yazılım gereksiniminin temelleri başlığında son olarak sistem ve yazılım gereksinimleri karşılaştırılması yapılmaktadır. Sistem, belirli bir amacı gerçekleştirmek üzere gerekli olan öğelerin ve bunlar arasındaki etkileşimin kombinasyonu olarak tanımlanmaktadır. Dolayısıyla sistem olarak isimlendirilen kavramın içine donanım, yazılım, insanlar, bilgi, yöntem, ortam, servisler ve diğer tüm tamamlayıcı öğeler de girmektedir.

Bilgi birikiminde gereksinim süreci 4 başlık altında incelenmektedir. Bu başlıklar sırasıyla süreç modeli, süreç aktörleri, süreç desteği ve yönetimi, süreç kalitesi ve iyileştirme şeklinde ifade edilmektedir (Bourque and Fairley, 2014). Gereksinim süreci, gereksinimlerin kim tarafından nasıl, ne zaman, ne şekilde edinileceğini, nasıl analiz edileceğini, dokümantasyonunun nasıl yapılacağını ve doğrulanacağının tarif edilmesidir (Bourque and Fairley, 2014).

Gereksinimlerin edinimi başlığı altındaysa paydaşlar, dokümanlar ve diğer kaynaklardan gereksinimlerin nasıl toplanacağını ve edinileceğini ifade edilmektedir. Burada yer alan alt başlıklar; gereksinim kaynakları ve gereksinim edinimi teknikleri olarak belirtilmiştir (Bourque and Fairley, 2014). Bir başka bilgi birikimi olan BABOK da gereksinimlerin edinimini ayrıca bir bilgi alanı olarak ele almış ve gereksinimleri edinmek için hazırlanma, gereksinimlerin edinimi ve gereksinimlerin teyit edilmesi görevlerini birer başlık olarak detaylandırmıştır.

Gereksinim analizi başlığı, analistin edindiği gereksinimler üzerinde çalışma yaparak, aralarındaki çakışmaları tespit etmesi ve çözmesi, ortaya çıkartılacak yazılımın sınırlarının tespit ederek, kapsamı belirlemesini içerir. Ayrıca kapsamın belirlenmesi, belirlenen kapsamın yazılımın geliştirileceği kurumun diğer yazılımları ve yazılım projeleriyle etkileşiminin belirlenmesi ve paydaşlardan toplanan gereksinimlerin yazılıma dönüşecek şekilde detaylandırılması da gereksinim analizi olarak ifade edilmektedir (Bourque and Fairley, 2014).

Gereksinimlerin belirtimi başlığı, sistematik olarak geliştirilen, gözden geçirilen onaylanan gereksinim dokümanının oluşturulması olarak tanımlanmaktadır (Bourque and Fairley, 2014). BABOK (2015) ise gereksinim analizi ve belirtimini "gereksinimlerin analizi ve tasarım tanımı" isimli tek bir bilgi alanı altında toplamaktadır.

Gereksinimlerin doğrulanması başlığı altındaysa gereksinimlerin gözden geçirilmesi, prototipleme, modellerle doğrulama ve kabul testlerinin tanımlanmasına değinilmektedir (Bourque and Fairley, 2014). Tüm bu çalışma sonuçlarına ve bulgulara dayanarak, mevcut modellerle karşılaştırmaların sonucuna bakıldığında önerilen modelin yazılım projelerinde uygulanmasının proje başarısı ve ürün kalitesine önemli oranda katkıda bulunacağı düşünülmektedir.

2.1.2. “Yazılım Gereksinimi” Tanımı

IEEE 610.12-1990 standardı yazılım gereksinimini aşağıdaki şekilde tanımlamaktadır:

1. Bir problemi çözmek ya da bir amaca ulaşmak için kullanıcı tarafından ihtiyaç duyulan durum (condition) ya da kabiliyet (capability),
2. Bir kontrat, şartname ya da diğer resmi nedenler dolayısıyla bir sistem ya da sistem bileşeni tarafından yerine getirilmesi zorunlu olan durum ya da kabiliyet,
3. 1. ve 2. maddelerdeki durum ya da kabiliyetin yazılı olarak ifade edilmesi.

IIBA’in BABOK kılavuzunun 2. sürümü yukarıdaki tanımı referans almakta ve tanıma düzenleyici (regülatör) paydaşını da ekleyerek şu şekilde derlemektedir: “Bir paydaş tarafından (paydaşlara kanun, sözleşme, şartname, standart dokümanı gibi bağlayıcı unsurlar da dahildir), belirli bir problemi çözmek ya da belirli bir amacı gerçekleştirmek üzere ihtiyaç duyulan durum ya da yetenek” (BABOK v2).

Yine BABOK’un 3. sürümündeyse gereksinim basitçe “bir ihtiyacın kullanılabilir ifadesi” şeklinde tanımlanmaktadır (BABOK v3).

Sommerville ve Sawyer (1997) “Requirements Engineering: A Good Practice Guide” isimli kitaplarında şu ifadeleri kullanmaktadırlar:

- Gereksinim, geliştirilmesi gerekenin ne olduğunun belirtimidir.
- Sistemin nasıl davranması gerektiğiyle ilgili açıklamalardır.
- Sistem özelliği ya da niteliğidir.
- Sistemin geliştirme süreciyle ilgili kısıtlar olarak karşımıza çıkabilir.

Wiegers (2003) “More About Software Requirements” isimli kitabında Sommerville ve Sawyer’ın tanımı üzerinden gitmekte ve şunları eklemektedir:

- Gereksinim, proje tamamlandığında sahip olacağımız şeydir. Sistemin nasıl davranması gerektiğiyle ilgili açıklamalar işlevsel gereksinimleri işaret eder.

- Gereksinimler aynı zamanda sistemin özelliklerinin ya da niteliklerinin açıklamaları da olabilir. Bu özellikler ve nitelikler sistemin ne yaptığından ziyade ne kadar iyi yaptığını tanımlamaya yöneliktir.
- Diğer bir gereksinim türüyse geliştirme sürecine etki eden kısıtlar olabilir. Özellikle teknolojiyle ilgili kısıtlar (yazılım dili, platform, veritabanı bileşenleri) ve kullanıcı arayüzü standartları en çok karşılaşılan süreç kısıtlarından ikisi olarak karşımıza çıkmaktadır.

Aurum ve Wohlin (2005), "Engineering and Managing Software Requirements" isimli kitaplarında gereksinimlerin sadece kullanıcılardan değil aynı zamanda kurumun genelinden, devletten ve endüstri standartlarından da geldiğinin altını çizmektedirler. Ayrıca ideal olarak gereksinimlerin yazılımın çözüm tasarımından bağımsız olması gerektiğini ancak bunun pratikte çok mümkün olmadığını ifade etmişlerdir.

Diğer taraftan Withall (2007), "Software Requirement Patterns" kitabında gereksinimin sistemin ne yapacağını tanımladığını, nasıl yapacağını tanımlamadığını vurgulamaktadır: "Gereksinim, çözülmesi gereken problemi ortaya koyar ve sistemin ne için var olduğu ve amacına ulaşmak için ihtiyaç duyduğu her şeyi tanımlar."

Literatürde yazılım gereksinimin en basit ve kısa tanımlarından biri Robertson ve Robertson (2006) "Mastering the Requirements Process" isimli kitaplarında yapılmıştır: "Gereksinim, ürünün yapması gereken bir şey ya da sahip olması gereken kalitedir."

Kulak ve Guiney (2003) ise şöyle bir tanımlama yapmışlardır: "Gereksinim, bir bilgisayar uygulamasının kullanıcıları için yapması gereken şeydir. Bu bir sistemin var oluşunu hak etmesi için sağlamak zorunda olduğu belirli bir fonksiyon, özellik, kalite ya da prensip olabilir."

SWEBOK (Bourque and Fairley, 2014) ise gereksinimi "Yazılım gereksinimi en temelinde, gerçek dünyadaki bir sorunu çözmek için sergilenmesi gereken bir özelliktir" şeklinde tanımlamaktadır. Kılavuz "yazılım" unsurunu esas aldığı için, yukarıdaki cümlede bahsedilen problemin yazılımlar tarafından çözümlenecek konular olduğunun altını çizmekte ve tanımı biraz daha detaylandırmaktadır: "Yazılım gereksinimi belirli, bir sorunu çözmek için geliştirilen ya da adapte edilen bir yazılım tarafından sergilenmesi gereken bir özelliktir."

Kruchten (2003) “Rational Unified Requirements” kitabında gereksinimi şu şekilde tanımlamaktadır: “Gereksinim bir sistemin sahip olması gereken durum ya da yetenektir.”

Alexander ve Stevens’a (2002) göre gereksinim: “Gereksinim, kullanıcıların ve diğer paydaşların isteklerini ortaya koyan ihtiyaç ifadesidir.”

Tüm bu tanımlar ele alındığında şu bulgulara ulaşılmaktadır:

- Gereksinimlerin kaynağı paydaşlardır. Paydaşlar sistemin kullanıcıları olabileceği gibi, aktif kullanıcı olmayan diğer bireyler ve birey olmayan kanun, standart, sözleşme gibi hükmedici şeyler de paydaş olarak veya gereksinim kaynağı olarak düşünülebilir.
- Belirli bir problemi çözmek ya da belirli bir amacı gerçekleştirmek üzere ihtiyaç duyulan durum ya da yetenektir.
- Yazılımın nasıl davranması gerektiğiyle ilgilidir.
- Yazılımın sahip olması gereken özellik ve niteliklerdir.
- Yazılımın geliştirme sürecini ilgilendiren kısıtlardır.
- Çözülmesi gereken problemi ortaya koyar.
- Yazılımın ne için var olduğu tanımlar.
- Yazılımın sahip olması gereken kalitedir.
- Yazılımın kullanıcıları için yapması, yerine getirmesi gereken şeydir.
- Yazılımda yer alan ihtiyaçlar ve kısıtlardır.
- Gereksinimin, çözüm tasarımından bağımsız olması idealdir.

Kısaca özetlenirse, literatürde gereksinimin ne olduğunu ifade etmek üzere herkesçe kabul edilmiş tek bir tanım bulunmamaktadır. Ancak ağırlıklı olarak üzerinde anlaşılan bazı noktalar bulunmaktadır. Bunlar şöyle sıralanabilir:

1. En temel amacımız; bir problemi çözmek ya da bir amacı gerçekleştirmektir. Bunu “iş ihtiyacı” kavramıyla ifade edilebilmektedir,
2. Gereksinimlerin kaynağı; paydaşlardır,

3. Gereksinim, ihtiyaç duyulan yetenekler, durumlar, şartlar, özellikler, nitelikler, işlevselliktir.

Bunun sonucunda gereksinim tanımı için şöyle bir özet yapılabilir: “Gereksinim, bir iş ihtiyacını karşılamak üzere paydaşların ihtiyaç duyduğu yetenekler ve özelliklerdir.”

Yazılım gereksinimleri çeşitli sınıflara ayrılmakta, belirli niteliklerle tanımlanmakta, paydaşlardan toplanmakta ve belirli bir hedefi gerçeklemeyi amaçlamaktadır.

2.1.3. Yazılım Gereksinimlerinin Sınıflandırılması

Yazılım gereksinimleriyle çalışırken bilinmesi gereken önemli konulardan biri de yazılım gereksinimlerinin sınıflandırılması olarak karşımıza çıkmaktadır.

Aurum ve Wohlin (2005) gereksinimleri sınıflandırırken farklı perspektiflere göre gruplayarak ele almışlardır:

Çizelge 2.1.3.1. Gereksinim Sınıfları

A	İşlevsel gereksinimler	Sistemin yapacağı “şey”dir.
	İşlevsel olmayan gereksinimler	Performans, güvenlik, modülerlik gibi işlevsel gereksinimleri yerine getirecek sistem üzerindeki kısıtlar.
B	Amaç seviyesindeki gereksinimler	İş hedefleriyle ilişkilidir.
	Problem alanı seviyesindeki gereksinimler	Problem alanıyla ilişkilidir.
	Ürün seviyesindeki gereksinimler	Ürünle ilişkilidir.
	Tasarım seviyesindeki gereksinimler	Üretilecek yazılımla ilgilidir.
C	Birincil gereksinimler	Paydaşlar elde edilen gereksinimlerdir.
	Türetilmiş gereksinimler	Birincil gereksinimlerden türetilmiş gereksinimlerdir.

D	İş gereksinimleri	İş alanı, iş hedefleriyle ilgili gereksinimler.
	Teknik gereksinimler	Çözüm için kullanılacak teknolojik alt yapı, programlama dili, platform gibi teknik gereksinimler.
E	Ürün gereksinimleri	İş ihtiyaçları ve kullanıcı ihtiyaçlarıdır.
	Süreç gereksinimleri	Süreçle ilgili gereksinimlerdir.

Gereksinim sınıflandırmasına Çizelge 2.1.3.1. de bulunan "E" grubundaki "proje gereksinimleri" başlığı altında yeni bir sınıf daha açılabilir. Böylece "E" grubunun genişletilmiş hali şöyle olabilir:

1. Ürün gereksinimleri,
2. Süreç gereksinimleri,
3. Proje gereksinimleri.

Burada proje gereksinimleri iş alanındaki süreç gereksinimlerini yerine getirecek ürünü ortaya koymak üzere yapılacak olan projenin, o projeyle ilgili gereksinimlerini ortaya koyacaktır.

Young (2004) bunlara ek olarak "belirtilen gereksinimler" ve "gerçek gereksinimler" diye adlandırdığı bir ayrıma daha gitmektedir. Bu ayrımda ifade edilen belirtilen gereksinimler, paydaşlarca ortaya koyulan gereksinimlerdir. Gerçek gereksinimler ise analiz boyunca olgunlaştırılan ve doğrulanan gereksinimleri ifade eder. Bu gereksinimler yine en başta paydaşların ortaya koyduğu gereksinimler olabileceği gibi, sonradan analist tarafından ortaya çıkartılmış gereksinimler de olabilmektedir.

Yine Young (2004) başka bir bakış açısıyla, gereksinimleri gereksinim analizi aşamasında bulundukları fazlara göre de sınıflandırmakta ve böylece doğrulanmış gereksinimler, geçerlenmiş gereksinimler gibi kavramları da gereksinim sınıflandırması olarak ortaya koymaktadır.

Yazılım gereksinimlerinin sınıflandırılmasındaki en önemli konulardan biri de, yazılımın kaliteye yönelik karakteristik özellikleriyle işleve yönelik gereksinimlerini iki farklı unsur olarak ele alan işlevsel ve işlevsel olmayan gereksinimler ayrımıdır.

2.1.4. Yazılım Gereksinimlerinin Seviyelendirilmesi

Yazılım gereksinimlerini sınıflandırırken dikkate alınması gereken bir başka unsur da gereksinimin seviyesidir. Yazılım gereksiniminin seviyesi, aynı zamanda yazılım gereksinimi sürecindeki aşamayla da ilgilidir. Gereksinim toplamanın ilk aşamalarında üst düzey gereksinimler toplanmaktadır. Bunlar "iş gereksinimleri" olarak isimlendirilmektedirler. Gereksinim toplama sürecinin ilerleyen aşamalarında her düzeyden paydaşla görüşülerek daha detaylı gereksinimler toplanır. Bunlar "paydaş gereksinimleri" olarak isimlendirilir. Son olarak analistin analiz görevini de yürütmesiyle birlikte ortaya "çözüm gereksinimleri" çıkmaktadır (BABOK v3).

Bazı araştırmacılara göre her seviyede işlevsel ve işlevsel olmayan gereksinimler toplanabilmekte ve her seviyenin sonunda ortaya çıkan doküman farklı içerik ve detaya sahip olabilmektedir (Bittner & Spence, 2002; Wiegers, 2003).

2.1.5. Yazılım Gereksinimlerinin Özellikleri

Literatürde yazılım gereksinimleri üzerine yapılan çalışmalardan biri de iyi bir yazılım gereksiniminin nasıl olması gerektiği üzerinedir. Bahsi geçen konu bazı kaynaklarda "mükemmel gereksinimin karakteristik özellikleri", bazı kaynaklarda "iyi biçimlendirilmiş gereksinim özellikleri (Wiegers, 2003)", bazı kaynaklarda da "iyi bir gereksinimin kriterleri (Young, 2004)" başlığı altında incelenmektedir.

Alexander ve Stevens (2002) "Writing Better Requirements" isimli kitaplarında iyi gereksinim için yönergeler başlığı altında şunlara değinmektedirler:

- Basit, anlaşılır cümleler kurun,
- Herkesin anlayabileceği kelimeler kullanın,
- Her bir gereksinim için o gereksinimi isteyen kullanıcıları belirleyin,

- Sonuçları ifade etmeye odaklanın,
- Kabul kriterlerini belirleyin,
- Muğlaklıktan kaçının,
- Birden fazla gereksinimi içeren gereksinimler yazmayın,
- Anlamca genişletilmeye müsait cümleler kurmayın,
- Çok uzun cümleler kurmaktan kaçının,
- Sistemi tasarlamayın,
- Gereksinimlerle tasarımı birbirine karıştırmayın,
- Gereksinimleri ve planlamayı birbirine karıştırmayın,
- Gereksinimler üzerinde spekülasyon yapmayın,
- Olasılık içeren gereksinimler yazmayın.

IEEE Guide for Developing System Requirements Specifications'a (1998) göre iyi biçimlendirilmiş bir gereksinimin aşağıdaki bileşenlere sahip olması gerekmektedir:

- Yetkinlik/Yetenek: Paydaş tarafından ihtiyaç duyulan özellik ya da işlevdir,
- Durum/Koşul: Sistem için tanımlanmış bir yetkinlik için tanımlanmış olan durumları ifade etmektedir,
- Kısıtlar: Belirli bir yetenek ya da durum için konulabilecek alt ve üst limitler, kısıtlayıcı durumlardır.

Gereksinimlerin karakteristik özellikleri olarak tanımlanabilecek bir diğer gereksinim nitelendirme yaklaşımı "Software Requirements" (Wiegers, 2003) isimli kitapta gereksinimler tekil olarak ve bir bütün olarak kişi şekilde sınıflandırılmıştır. Bu yaklaşımda her bir gereksinim tek tek ele alındığında;

- tamamlanmış (complete),
- doğru (correct),
- uygulanabilir (feasible),
- gerekli (necessary),

- önceliklendirilmiş (prioritized),
- belirsizliğe yer vermeyen (unambiguous),
- gerçekliği kanıtlanabliir (verifiable),

olmalıdır. Ayrıca gereksinimler bir arada değerlendirildiklerinde;

- birbirini tamamlayan (complete),
- tutarsızlığa yer vermeyen (consistent),
- değiştirilebilir (modifiable),
- izlenebilir (traceable),

olması gerektiği belirtilmiştir. Ayrıca bu karakteristik özelliklere ek olarak aşağıdaki özellikler de literatürde yerini almıştır (Firesmith, 2003; Young, 2004; Wiegers, 2003; IEEE-STD 1233-1998).

- Gereksinimler emsalsiz (unique) olmalıdır,
- Normalleştirilmiş (normalized) olmalıdır,
- Sınırlandırılmış (bounden) olmalıdır,
- Versiyonlandırılmış (configurable) olmalıdır,
- Yeterince detaylandırılmış (granular) olmalıdır,
- Özlü ve kısa (concise) olmalıdır,
- Sistem bileşenleriyle eşleştirilmiş (allocated) olmalıdır.

Bunların dışında yazılım gereksinimlerinin kaynağı, öncelik durumu gibi ek bilgilere sahip olması gerekir ve bunlara gereksinimin niteliği (requirements attribute) adı verilmektedir. Her bir gereksinim için belirlenebilecek bu nitelikler gereksinim belirteci (identification), öncelik, kritiklik seviyesi, uygulanabilirlik, risk, kaynak (paydaş), tip şeklinde olabilmektedir (IEEE-STD 1233-1998).

2.1.6. İş Kuralları

İşlevsel gereksinimin temel öğelerinden biri olmasına rağmen iş kuralları önemi gereği bu başlık altında ayrıca ele alınmıştır.

İş kuralları işlevsel fonksiyonların temellerini oluştururlar. İş kurallarını iki grup altında toplamak mümkündür (Young,2004):

- Sistem tarafından desteklenecek iş aktiviteleriyle ilgili politikalar, durumlar, koşullar ve kısıtlar,
- İşlevsel gereksinimin arkasındaki karar verme kılavuzu, yöntem ve kontroller,
- İş süreciyle ilgili tanımlar,
- İş sürecindeki iş akışı ve unsurlar arasındaki ilişkiler,
- İşi yapmak için gerekli olan bilgi.

Business Rules Group iş kuralını "İşle ilgili bazı kavramları kısıtlar ya da tanımlar" olarak ifade etmektedir. İş kurallarıyla ilgili birçok sınıflandırma önerisi bulunmakla birlikte aşağıdaki yapı basit bir şekilde iş kurallarını sınıflandırmaktadır (Wiegers, 2003).

- İşle ilgili olgular, unsurlar, durumlar, gerçekler (facts),
- İşle ilgili limitler, kısıtlar (constraints),
- İşle ilgili eylem başlatıcılar (action enablers),
- İşle ilgili hesaplamalar (computations),
- İşle ilgili çıkarımlar, sonuçlar (inferences).

Her bir gereksinim mutlaka bir iş kuralına dayanmıyor gibi görünse de aslında arkasında bir iş bilgisini, unsurunu, bir kısıdı, bir eylem başlatıcısını, bir hesaplamayı, bir çıkartım ve sonucu ya da bunların kombinasyonlarını içermektedir.

2.1.7. Gereksinim Kaynakları ve Paydaşlar

Başarılı bir yazılım projesi için doğru gereksinimler önemli olduğu gibi, gereksinimlerin doğru olabilmesi için gereksinim kaynaklarının doğru tespit edilmesi ve iyi yönetilmesi de önemlidir. Paydaş en basit tanımıyla gereksinimlerin kaynağıdır. Sistemin kullanıcıları doğal olarak paydaş gruplarından biri olmakla birlikte sistemi direkt kullanmayacak olsa da parasını ödeyen destekleyiciler, başarısından sorumlu olan karar vericiler, çalışmasını sağlayacak olan işletim personeli, desteğini verecek olan yardım masası çalışanları ve eğer ilişkiliyse kanunlar ve sözleşmeler de paydaş olarak görülmektedir (Alexander & Stevens, 2002).

Bir başka kaynak olan BABOK paydaşı, yapılacak olan yazılımı etkileyen ya da yazılımdan etkilenen herkes olarak tanımlamaktadır. Yazılım gereksinimlerinin edinilmesi aşamasında iş analistinin göz önünde bulundurması gereken gereksinim kaynakları için SWEBOK (2014), yazılımın amacını, iş alanı bilgisini, paydaşları, iş kurallarını, işletim ortamını ve kurum ortamını göstermektedir. Her iki kaynağın kesişim kümesi olarak gereksinim kaynakları aşağıdaki şekilde listelenebilmektedir:

- Müşteri,
- İş alanı uzmanı,
- Son kullanıcı,
- Tedarikçi,
- Uygulama uzmanı,
- Proje yöneticisi,
- Test uzmanı,
- Kanun, kontrat, sözleşme,
- Destekleyici.

Yukarıdaki tanımlamaların haricinde son kullanıcılar ve paydaşların tanımlamalarıyla ilgili bazı ek sınıflandırmalar da yapılabilmektedir. Örneğin sistemin kullanıcılarını temsilen seçilecek ve diğer kullanıcılara göre yaptığı işle ilgili alan bilgisi,

bilgisayar kullanım becerisi ve yazılım proje deneyimi daha fazla olan kişilere anahtar kullanıcı denilmektedir (Alexander & Beus-Dukic, 2009; Wiegers, 2003).

Bu konuyla ilgili bir başka deyim de "ürün şampiyonu" olarak tanımlanmaktadır.. Ürün şampiyonu ya da şampiyonları geliştirilecek sistemle ilgili iş bilgisine ve deneyime sahip paydaşları temsil etmektedir. Gereksinimlerin büyük çoğunluğu bu paydaşlardan alınmaktadır (Alexander & Beus-Dukic, 2009).

2.1.8. Gereksinim Mühendisliği Ölçütleri

Geliştirilecek olan ürünün ve bu ürünü geliştirmek için harcanacak emeğin, dolayısıyla da gerekli bütçenin belirlenmesi açısından gereksinimlerin büyüklüğünün ölçülmesi önemli bir faaliyettir (Bourque & Fairlay, 2014). Bir yazılımın işlevsel büyüklüğüyle ilgili meta-standart tanımlamalarından biri ISO 14143-1 olarak belirlenmektedir. Bunun dışında ISO tarafından belirlenen ISO 19761- COSMIC, ISO 20926-IFPUG, ISO 20968-MKII, ISO 24570-NESMA ve ISO 29881–FISMA standartları da yazılımın işlevsel büyüklüğünü ölçmek üzere metotlar sunmaktadır.

Boehm, Abts ve Chulani (2000), yazılım geliştirme maliyet tahminleme üzerine yaptıkları çalışmada tahminleme teknikleri 6 kategoriye ayırmaktadırlar:

- Model tabanlı teknikler,
- Uzmana dayalı teknikler,
- Öğrenmeye dayalı teknikler,
- Dinamik teknikler,
- Regresyona dayalı teknikler,
- Kompozit teknikler.

Model tabanlı tekniklerden Software Life-Cycle Model (SLIM) yazılım proje yaşam döngüsünde personel seviyesinin zamana göre dağılımına dayanmaktadır. Yine model tabanlı tekniklerden bir başkası da "Checkpoint" olarak geçmektedir. Sofware Productivity Research (SPR) tarafından geliştirilen bu teknikte işlev puanlama esas alınmıştır. Checkpoint'e benzer şekilde işlevlere dayalı olarak çalışan diğer teknikler

COSMIC (Common Software Measurement International Consortium), IFPUG, MkII, NESMA ve FFP olarak sayılabilir. Model tabanlı diğer tekniklerden bazılarıysa PRICE-S, ESTIMACS, SEER-SEM, SELECT Estimator, COCOMO II (COnstructive COst MOdel) olarak geçmektedir. Uzmanlığa dayalı tahminleme tekniklerinden en bilineni Delphi Tekniği olarak geçmektedir. Bunun haricinde literatürde geçen tekniklerde bir diğeri de Work Breakdown Structure (WBS)'dir (Boehm, Abts, Chulani, 2000).

Öğrenmeye dayalı teknikler kategorisinde "Case Study", "Neural Networks" başlıca yaklaşımlarken, dinamik teknikler içerisinde "Systems Dynamics" öne çıkmaktadır. Regresyon temelli tekniklerde öne çıkanlar "Standard", "Regression – Ordinary Least Squares (OLS)", "Robust" ve kompozit teknikler olarak da "COCOMO" ve "COCOMO II" sayılabilmektedir (Boehm, Abts, & Chulani, 2000; Fehlmann & Santillo, 2010; Kemerer, 1987).

Yukarıda sayılan bu teknikler daha çok çevik yöntemlerin ortaya çıkması ve kullanılması öncesinde geliştirilen, geleneksel yöntemlerle geliştirilen yazılımlara odaklı yaklaşımlardır. Çevik yöntemler için daha çok "Story Point" ve "Use Case Point" yaklaşımları kullanılmaktadır (Georgsson, 2011).

2.2. Gereksinim Analizi Süreci

Yazılım gereksinim analiz süreci ve gereksinimlerin yönetimi en az gereksinimlerin geliştirilmesi kadar önemli olduğu için ayrı bir başlık altında ele alınmıştır. Gereksinim analizi, kullanıcı ihtiyaçlarını sistemin donanım ya da yazılım gereksinimlerine dönüştürmek ya da mevcut gereksinimlerin iyileştirilmesi olarak tanımlanmaktadır (IEEE-STD 610.12-1990; Firesmith, 2004).

Aurum ve Wohin (2005) gereksinimlerin yönetimi için yapılması gereken uygulamaları 5 başlık altında gruplandırmaktadırlar:

1. Gereksinimlerin edinilmesi, belirtimi ve modellenmesi,
2. Önceliklendirme,
3. Gereksinimlerin bağımlılıkları ve etki analizi,

4. Gereksinimlerin müzakeresi,
5. Kalite güvence.

Gereksinim mühendisliği Şekil 2.1.1.1 de de görülebileceği üzere, gereksinimlerin geliştirilmesi ve gereksinimlerin yönetimi olarak iki ana başlık altında incelenmekte ve gereksinim geliştirme başlığı altında da gereksinimlerin edinilmesi, analiz, belirtim ve doğrulama görevlerine değinilmektedir.

Alexander ve Beus-Dukic (2009) "Discovering Requirements" isimli kitaplarında gereksinim analizini en basit süreç haliyle gereksinimlerin keşfi, yazılı hale getirilmesi ve doğrulanması şeklinde tarif etmektedirler.

Biraz daha detaylı bir yaklaşımla bir gereksinim analiz yaşam döngüsünde gerçekleştirilmesi gereken aktiviteler aşağıdaki gibidir (Young, 2004):

- Paydaşların belirlenmesi,
- Gereksinimlerin edinilmesi,
- Gereksinimlerin tespiti,
- Gereksinimlerin açıklığa kavuşturulması ve yeniden ifade edilmesi,
- Gereksinimlerin analizi,
- Gereksinimlerin tüm paydaşlar için aynı anlama gelecek şekilde tanımlanması,
- Gereksinimlerin belirtimi,
- Gereksinimlerin önceliklendirilmesi,
- Gereksinimlerin türetilmesi,
- Gereksinimlerin bölümlenmesi,
- Gereksinimlerin alt sistem ve bileşenlerle örtüştürülmesi,
- Gereksinimlerin izlenmesi,
- Gereksinimlerin yönetilmesi,
- Gereksinimlerin sınanması ve doğrulanması,
- Gereksinimlerin geçerlenmesi.

BABOK (v3) gereksinimleri ortaya çıkartmak için yapılması gereken aktiviteler bilgi alanı adı verilen süreç benzeri yapılardan oluşan gruplarla ifade edilmiştir. Bir iş analistinin yazılım gereksinimlerini geliştirmesi ve yönetmesi için aşağıdaki görevleri yerine getirmesi gerektiğini ifade etmektedir:

- İş Analizinin Planlanması ve İzlenmesi;
  - İş Analizi Yaklaşımının Planlanması,
  - Paydaşların Angajmanının Planlanması,
  - İş Analizi Yönetiminin Planlanması,
  - İş Analizi Bilgi Yönetiminin Planlanması,
  - İş Analizi Performans İyileştirme Fırsatlarının Tespiti,
- Gereksinimlerin Edinilmesi ve Birlikte Çalışma;
  - Gereksinimlerin Edinimi Öncesi Hazırlanmak,
  - Gereksinimlerin Edinilmesi,
  - Toplanın Gereksinimlerin Teyidinin Alınması,
  - İş Analizi Bilgilerinin İletişiminin Yapılması,
  - Paydaşlarla İşbirliğinin Yönetilmesi,
- Gereksinim Yaşam Döngüsü Yönetimi;
  - Gereksinimlerin İzlenmesi,
  - Gereksinimlerin İdame Ettirilmesi,
  - Gereksinimlerin Önceliklendirilmesi,
  - Gereksinim Değişikliklerinin Değerlendirilmesi,
  - Gereksinimlerin Onaylanması,
- Strateji Analizi;
  - Mevcut Durumun Analizi,
  - Gelecekte Beklenen Durumun Tanımlanması,

    - Risklerin Değerlendirilmesi,
    - Değişiklik Stratejisinin Tanımlanması,
- Gereksinimlerin Analizi ve Tasarımın Tanımlanması;
    - Gereksinimlerin Belirtimi ve Modellenmesi,
    - Gereksinimlerin Doğrulanması,
    - Gereksinimlerin Geçerlenmesi,
    - Gereksinim Mimarisinin Tanımlanması,
    - Tasarım Seçeneklerinin Tanımlanması,
    - Potansiyel Değerin Analiz Edilmesi ve Çözüm Önerilmesi,
- Çözümün Değerlendirilmesi;
    - Çözümün Başarımının Ölçülmesi,
    - Başarım Ölçütlerinin Analiz Edilmesi,
    - Çözüm Limitlerinin Değerlendirilmesi,
    - Kurumsal Limitlerin Değerlendirilmesi,
    - Çözümün Değerini Artıracak Aksiyonların Önerilmesi.

2.2.1. Yazılım Gereksinimlerinin Edinilmesi

BABOK (v3), gereksinimlerin edinilmesi aktivitesini, yapılacak olan değişiklik (proje ya da değişiklik talebi) ile ilgili bilgilerin belirlenmesi, keşfedilmesi ve aydınlatılması olarak tanımlamaktadır.

Yine BABOK'da (v3) gereksinimleri edinme aktivitelerini ortak çalışmaya dayalı, araştırmaya dayalı ve deneyime dayalı olmak üzere 3 başlık altında incelemektedir ve gereksinimlerin edinilmesi için 18 adet teknik tanımlamaktadır. Araştırmaya dayalı gereksinim edinmede daha önce kayıtlı belge ve bilgiler esas alınmaktadır. Deneyime dayalı gereksinim edinme ise gözlem çalışmaları, prototipleme ve kavram kanıtlama olarak karşımıza çıkmaktadır.

"Discovering Requirements" isimli kitaplarında Alexander ve Beus-Dukic (2009) gereksinim edinme tekniklerini bireylerden edinilen gereksinimler, gruplardan edinilen gereksinimler ve "insan dışı unsurlardan" alınan gereksinimler olarak 3 gruba ayırmaktadır. Burada "insan dışı unsur" ile kastedilen prototipleme, tersine mühendislik, gereksinimlerin yeniden kullanımıdır.

Literatürdeki çeşitli kaynaklar tarandığında gereksinim edinme teknikleri olarak şu liste derlenebilmektedir:

- Kıyaslama,
- Pazar analizi,
- Beyin fırtınası,
- İş kuralı analizi,
- İş birliği oyunları,
- Kavram modelleme,
- Veri madenciliği,
- Veri modelleme,
- Doküman analizi,
- Odak grup çalışması,
- Arayüz analizi,
- Bire-bir görüşmeler,
- Zihin haritalama,
- Yerinde gözlemleme,
- Süreç analizi,
- Süreç modelleme,
- Prototipleme,
- Anket,
- Çalıştay,

- Tersine mühendislik,
- Sorun/hata/öneri bildirimleri.

2.2.2. Yazılım Gereksinimlerinin Analizi

Yazılım gereksinimlerinin analizi, edinilen gereksinimlerin arasındaki uyuşmazlığın tespiti ve düzeltilmesi, yazılımın sınırlarının tespitiyle birlikte kurumsal ve işletim ortamlarıyla nasıl etkileşime gireceğinin belirlenmesi, yazılım gereksinimlerinin ortaya çıkması için gereksinimlerin detaylandırılması şeklinde tanımlanmaktadır (Bourque and Fairley, 2014).

Gereksinim analizinin alt fonksiyonları aşağıdaki gibi listelenmektedir:

- Gereksinimlerin işlevsel ve işlevsel olmayan gereksinimler şeklinde sınıflandırılması,
- Gereksinimlerin birbirleriyle olan ilişkilerinin belirlenmesi (hiyerarşisi ve izlenebilirlik matrisinin oluşturulması),
- Gereksinimlerin ürün ya da süreç gereksinimi olup olmadığının belirlenmesi,
- Gereksinimlerin önceliklerinin belirlenmesi,
- Gereksinimlerin kapsam içinde yönetilmesi,
- Gereksinimlerin kararlı hale gelmesinin ya da kararlılık durumlarının belirlenmesi,
- Kavramsal modellemenin yapılması,
- Mimari tasarımla gereksinimlerin eşleştirilmesi,
- Gereksinimlerin müzakeresi,
- Formal analiz.

2.2.3. Yazılım Gereksinimlerinin Belirtimi

Yazılım gereksinim bildirimi, yazılım gereksinimlerinden oluşan, sistematik olarak gözden geçirilen, değerlendirilen ve onaylanan bir doküman oluşturmaktır. Belirtim olarak SWEBOK'da sistem tanımı dokümanı, sistem gereksinimleri dokümanı ve yazılım gereksinimleri dokümanları kastedilmektedir (Bourque and Fairley, 2014).

IEEE'nin yazılım gereksinimleri belirtimi için yayınladığı 830-1998 numaralı standarda göre bir yazılım gereksinim belirtimi belirli birtakım işlemleri yerine getiren belirli bir yazılım ürünü, program ya da programlar kümesi için oluşturulan dokümandır. 830 numaralı standarda göre iyi bir yazılım gereksinim belirtimi üretebilmek için aşağıdaki konulara eğilmek gerekmektedir:

- Gereksinimlerin mahiyetine değinmelidir. Gereksinimle ilgili işlevsellik, harici ara birimler, performans, nitelikler ve tasarım bu başlık altında ele alınmaktadır,
- Çevre birimleri ve unsurları ele almalıdır,
- Gereksinimler doğru, muğlak olmayan, tamamlanmış, tutarlı, önceliklendirilmiş, doğrulanabilir, değiştirilebilir ve izlenebilir olmalıdır,
- Doküman müşteri ve proje ekibiyle birlikte hazırlanmalıdır,
- Gereksinimler sürekli olarak detaylanıyor, netleşiyor ve değişiklikler iyi yönetiliyor olmalıdır,
- Gereksinimler dokümante edilirken prototipleme tekniği kullanılarak müşteri geri bildirimleri alınmalıdır,
- Kavramsal tasarım dokümana dahil edilmelidir,
- Proje gereksinimleri dokümana dahil edilmelidir.

Yukarıdaki doküman hazırlama önerileriyle birlikte IEEE bir analiz dokümanında olması gereken başlıkları şöyle tanımlamaktadır (IEEE-STD 830-1998):

- Giriş,
    - Amaç,
    - Kapsam,

        - Tanımlar, kısaltmalar,
        - Referanslar,
        - Genel açıklamalar,
    - Genel tanım,
        - Ürün perspektifi,
        - Ürün fonksiyonları,
        - Kullanıcı karakteristikleri,
        - Kısıtlar,
        - Varsayımlar ve bağımlılıklar,
    - Özel gereksinimler,
    - Ekler,
    - İndeks.

IEEE'den başka yazılım dokümanlarını tanımlayan bir başka doküman da MIL-STD-498 (1998) olarak görülmektedir. MIL-STD-498 hem yazılım geliştirme hem de dokümantasyonunu tarif eden bir standart olarak DI-IPSC-81433 referans numarasıyla dokümanın başlıklarını şu şekilde tanımlamaktadır:

- Kapsam,
- Referans alınan dokümanlar,
- Gereksinimler;
    - Gereksinim duyulan durumlar,
    - Sistem yetkinlik gereksinimleri,
    - Harici arayüz gereksinimleri,
    - Dahili arayüz gereksinimleri,
    - Dahili veri gereksinimleri,
    - Adaptasyon gereksinimleri,
    - Emniyet gereksinimleri,

        - Güvenlik ve gizlilik gereksinimleri,
        - Çevre gereksinimleri,
        - Bilgisayar kaynak gereksinimleri,
        - Yazılım kalite faktörleri,
        - Tasarım ve uygulama kısıtları,
        - Proje ekibiyle ilişkili gereksinimler,
        - Eğitimle ilişkili gereksinimler,
        - Lojistikle ilişkili gereksinimler,
        - Diğer gereksinimler,
        - Paketleme gereksinimleri,
        - Gereksinimlerin önceliği ve kritikliği,
    - Yeterlilik Koşulları,
    - Gereksinimlerin izlenebilirliği,
    - Notlar,
    - Ekler.

Analiz dokümanın genel olarak tanımlandığı bu yaklaşımların dışında BABOK (v3) gereksinimlerin her birinin teker teker belirtimi için de tanım yapmakta ve gereksinimin belirtimiyle modellenmesi bir arada ele almaktadır ve gereksinimlerin belirtimi için metin, matris, model/diyagram ya da bunların kombinasyonlarının kullanılabileceğini ifade etmektedir.

Nuseibeh ve Easterbrook (2000) "Requirements Engineering: A Roadmap" isimli çalışmalarında modelleme ve gereksinim analizini bir başlık olarak ele almışlardır. Gereksinimlerin birbirleriyle ilişkisini, doğruluğunu, bütünlüğünü, tutarlılığını belirlemek adına bu yaklaşım doğru olmakla beraber, modelleme BABOK (v3) da ifade edildiği gibi aynı zamanda dokümantasyonun da önemli bir bileşeni olmaktadır. Burada modeller kategorilere ayrılmakta ve bu kategoriler insanlar ve roller, gerekçe, aktivite akışı, yetkinlik, veri ve bilgi olarak gruplara ayrılmaktadır.

Yine buna benzer şekilde Beatty ve Chen (2012) yazdıkları "Visual Models for Software Requirements" kitabında gereksinimleri modellerle analiz etmenin ve dokümantasyonunu yapmanın önemini vurgulamakta ve modelleri hedef modelleri, insan modelleri, sistem modelleri ve veri modelleri olarak gruplara ayırmaktadırlar.

Geleneksel yöntemlerle yazılım proje geliştirmede yukarıdaki yaklaşımlar öne çıkarken, çevik yaklaşımlarda kullanıcı hikayeleri ve kullanım vakaları gereksinim edinme tekniği olarak ele alınmaktadır. Çevik yaklaşımlarda gereksinim belirtimleri için zorunluluk olmamakla beraber çoğunlukla kullanıcı hikayeleri tercih edilmektedir (Firesmith, 2005).

2.2.4. Yazılım Gereksinimlerinin Doğrulanması

Gereksinimlerin doğrulanması, hem gereksinimlerin doğru anlaşılıp anlaşılmadığının geçerlenmesi hem de gereksinimlerin kurum standartlarına uygun olup olmadığının doğrulanması olarak ifade edilmektedir (BABOK v3).

Doğrulama (verification) aktiviteleri BABOK v3'de şu şeklide detaylandırılmaktadır:

- Kurumsal performans standartlarına göre gereksinim analizinin doğru araç ve metotlarla yapılıp yapılmadığının kontrolü,
- Doğru modellerin, notasyonun, şablonların ve formların kullanılıp kullanılmadığının kontrolü,
- Her bir modelin tamamlanmış olduğundan emin olma,
- Modellerin birbirleriyle ve kendi içlerinde karşılaştırılarak unsurların ve referansların uygunluğunun kontrolü,
- Gereksinimlerin açıkça anlaşılabilir olduğunun kontrolü

Geçerleme paydaşların ifade ettikleri gereksinimlerle, hedeflenen çözüme ulaşılıp ulaşılamayacağının ve belirtilen gereksinimlerin doğru anlaşılıp anlaşılmadığının doğrulanmasıdır (Bourque and Fairley, 2014).

2.3. Yazılım Gereksinimlerinin Yönetimi

Yazılım gereksinimlerinin yönetimi, SWEBOK'da (Bourque and Fairley, 2014) gereksinimlerin değişikliklerinin kontrolü, versiyon kontrolü, gereksinim durum takibi ve gereksinimlerin birbirleriyle ilişkisinin izlenmesi şeklinde tanımlanmaktadır.

Gereksinimlerdeki değişikliklerin yönetimi, yapılacak değişikliğin mevcut gereksinimlere ve proje öğelerine etkisinin analizini gerektirir (BABOK v3; Krishnamurthy & Saran, 2003). Değişiklik yönetimi aynı zamanda yazılım geliştirme metodolojisine de bağımlıdır. Eğer yazılım geleneksel yöntemle geliştiriliyorsa bu durumda değişiklik talepleri dokümantasyon ve onaylama mekanizmasına dayalı resmi bir süreçle yönetilir. Eğer yazılım çevik bir süreçle geliştiriliyorsa herhangi bir yineleme öncesinde, gerçekleştirilmesi beklenen gereksinimlerin değiştirilmesinde sakınca görülmemektedir.

Gereksinim versiyon kontrolü, gereksinimlerle ilgili çatışmaların ve açık konuların çözülmesini, paydaşların genel kabulünün kazanılmasını ve karar vericiler tarafından onaylanmasını kapsamaktadır (BABOK v3).

Gereksinimlerle etkili bir şekilde çalışabilmek için öz niteliklerin, önceliklendirmenin ve izlenebilirliğin en iyi şekilde yürütülmesi gerekmektedir.

Gereksinimlerin her birinin ayrı ayrı önerilmiş, kabul edilmiş, doğrulanmış, geçerlenmiş, ertelenmiş, iptal edilmiş, uygulanmış vb. öz niteliklerinin takip edilmesi, gereksinim yönetimi aktiviteleri içerisinde yerini almaktadır (Bourque and Fairley, 2014; BABOK v3).

Bu öz niteliklere ek olarak gereksinimlerin öncelik durumlarının takibi de önemli bir noktadır. Önceliklendirmenin en iyi şekilde yapılabilmesi için de bu sürecin, risklerinin ve gerekliliklerinin iyi anlaşılması ve uygulanması gerekmektedir (Firesmith, 2004).

Gereksinimlerin izlenebilirliği sürecinde ise yüksek seviyeli gereksinimlerden başlayarak analizin ilerleyen aşamalarında detaylanan gereksinimlerin birbirleriyle olan alt-üst, öncül-ardıl, birliktelik gibi ilişkilerinin takip edilmesi gerekmektedir. Doğru ve

başarılı ve etki analizi için gereksinimlerin birbirleriyle, tasarım öğeleri ve diğer çözüm bileşenleriyle ilişkilerinin izlenmesi çok önemlidir (BABOK v3).

Müşteri İlişkileri Yönetimi uygulaması Çizelge 2.3.1. de görüleceği üzere, Kurumsal Kaynak Planlama yazılımı perspektifinden bakıldığında bir modül gibi görünmekle beraber, Müşteri İlişkileri Yönetimi kendi başına ele alındığında bir sistem olarak görülmekte ve alt özellikleri birer alt sistem olarak karşımıza çıkmaktadır.

Çizelge 2.3.1. Kurumsal Kaynak Planlama ve Müşteri İlişkileri Yönetimi Sistem Modül Karşılaştırması

Kurumsal Kaynak Yönetimi Alt Sistemleri	**Müşteri İlişkileri Yönetimi Alt Sistemleri**
• Finans Yönetimi, • Hammadde Yönetimi, • Masraf/Harcama Yönetimi, • Proje Yönetimi ve Proje Muhasebesi, • Tedarik Zinciri Yönetimi, • Envanter Yönetimi, • Depo Yönetimi, • Sipariş Yönetimi,, • Üretim ve İmalat • İnsan Kaynakları Yönetim ve Bordrolama, • Demirbaş Yönetimi, • **Müşteri İlişkileri Yönetimi, Satış ve Pazarlama (CRM):** o Pazarlama Otomasyonu, o Satış Yönetimi, o Aktivite Takibi, o Telefonla Pazarlama, • Web Portali: • İş Akışları ve Bildirimler…	• Müşteri Hesapları, • İletişim Yönetimi, • Satış Fırsatı Yönetimi, • Potansiyel Müşteri Yönetimi, • Pazarlama Yönetimi, • Rakip Yönetimi, • Ürün Yönetimi, • Satış Literatürü, • Kota Yönetimi, • Sipariş Yönetimi, • Fatura Yönetimi, • Hizmet Takvimi Yönetimi, • Durum Yönetimi, • Bilgi Birikimi Yönetimi, • Kontrat Yönetimi, • Hizmet Yönetimi.

Literatürde ve sektörde modül olarak geçen bu kavramın önerilen modeldeki karşılığı alt sistemdir. Modelin sadeleştirilmesi için çok sayıda kavram ve bileşen yerine daha yalın bir yaklaşım sergilemek amacıyla modül yerine alt sistem kavramının kullanılması tercih edilmiştir.

"Özellik" (feature) için literatürde yapılan tanımlardan bazıları şöyledir:

Leffingwell ve Widrig özellik ile ilgili olarak iki tanım yapmışlardır: "Bir ya da daha fazla paydaşın ihtiyacını karşılamak üzere sistem tarafından sağlanan hizmet." "Sistemden beklenilen davranışların üst düzey ifadelerle ortaya konmuş haline bir ürün ya da sistemin özellikleri denir" (Leffingwell & Widrig, 1999).

Beatty ve Chen'in yaptığı açıklamaya göre de iki tanım bulunmaktadır: "Özellik, iş hedeflerini gerçekleştirmek üzere tanımlanan işlev alanını ifade etmektedir." "Özellikler, gereksinimleri ifade ve organize etmek için kullanılan gereksinim kümeleridir" (Beatty & Chen, 2012).

BÖLÜM 3

MEVCUT GEREKSİNİM BELİRTİMLERİNİN ve ANALİZ SÜREÇLERİNİN DEĞERLENDİRİLMESİ

Bu bölümde literatürde geçen ve sektörel uygulamalarda aktif olarak kullanılan mevcut yazılım gereksinim analizi süreçlerine değinilmektedir. Mevcut süreçler, belirtim ve modelleme yöntemleri birlikte göz önünde bulundurularak, avantaj ve dezavantajlarıyla birlikte tartışılmaktadır.

3.1. Analiz Süreçleri

Analiz süreci yazılım geliştirme sürecinin bir alt süreci olduğundan, analiz sürecini değerlendirirken büyük resim olan yazılım geliştirme süreçleri perspektifinden bakmak daha doğru olacaktır.

Yazılım gereksinim analizi süreçleri literatürde birçok kaynakta tanımlanmakla birlikte bu alanda en geniş tanımlamayı ve açıklamayı iş analizi çerçevesinde BABOK yapmaktadır. BABOK analiz süreci için genel bir tanımlama yaparak ortaya bir çerçeve koymakta, geleneksel ya da çevik yaklaşıma özel detaylı tarif içeren ayrıntılı bir süreç sunmamaktadır. Benzer şekilde SWEBOK (Bourque and Fairley, 2014) da bir iş analizi sürecinde olması gereken maddeleri ve maddelerde yerine getirilmesi gereken temel işlemleri ortaya koymaktadır.

Diğer yandan çevik süreç yaklaşımları ağırlıklı olarak yakın iletişimi öne koymakta ve ayrıntılı dokümantasyonun geliştirme aktivitelerini engellememesi gerektiği savını öne sürmektedirler. Özellikle çevik yaklaşımlara yönelik gereksinim analizinde kapsamlı bir dokümantasyon zorunlu değildir. Bunun yerine tüm ekip için ortak bir

referans noktası oluşturmaya yönelik kısa, işlev isimlerinden oluşan kullanıcı hikayeleri esas alınmaktadır. Geleneksel süreçlerde analiz görevi analist adı verilen uzmanlar tarafından gerçekleştirilirken, çevik süreçlerde analiz faaliyeti "ürün sahibi" adı verilen ve görev alanına analiz haricinde test, proje yönetimi, ürün yönetimi de giren farklı bir rol tarafından gerçekleştirilmektedir (Boyer & Mili,2011; Lucia & Qusef, 2010).

Analiz süreçleri genel olarak ele alındığında, analizde yapılan hataların önemli bir bölümünün ifade biçimi, dokümantasyon ve doğal dil kullanımı gibi iletişime dayalı faktörlere dayandığı görülmektedir (Al-Rawas1 & Easterbrook, 1996). Aşağıdaki başlıklarda mevcut analiz süreçleri olumlu ve olumsuz yönleri açısından incelenmektedir.

3.1.1. Geleneksel Analiz Süreçleri

Şelale ve Spiral gibi geleneksel yazılım geliştirme yaklaşımlarında yazılımın gerçekleştirebilmesi için gerekli olan analiz, mimari tasarım, geliştirme, stabilizasyon ve işletime alma fazları birbirini izleyen sırayla gerçekleştirilmektedir. Geleneksel yöntemin temel motivasyonu belirli bir kapsam çizebilmek ve yazılımı ortaya koyabilmek için gerekli olan proje planını yaparak, kapsamın dışına mümkün olduğu kadar çıkmadan projeyi tamamlamaktır (Withall, 2007).

Yukarıda ve bir önceki bölümde de bahsedildiği üzere geleneksel yazılım geliştirme yaklaşımında detaylı tarif edilmiş bir analiz süreci izlenmemektedir. Ancak kaynaklar birlikte değerlendirildiğinde yazılımın tüm aşamalarında gereksinim toplama görevinin çalıştığı, başlangıçta stratejik analizin, sonrasında paydaş analizinin ve en sonunda detaylı analizin yapıldığı görülmektedir.

3.1.2. Çevik Analiz Süreçleri

Günümüzdeki iş uygulamalarını sürekli değişen iş öncelikleri ve rekabet nedeniyle belirli bir kapsam içinde sabitlemek çok mümkün olamamaktadır. Bu nedenle çevik yaklaşım değişikliğe açık bir şekilde, belirli bir süre içinde ve belirli bir kaynakla/bütçeyle üretilebilecek en yüksek değeri üretmeyi hedeflemektedir. Çevik analiz süreci bunu yaparken bir önceki bölümde detaylıca anlatıldığı şekilde yakın

çalışmaya ve yüz yüze çalışmaya dayalı şekilde paydaşlarla etkileşim içerisinde analizi gerçekleştirmeye odaklanmıştır (Boyer & Mili, 2011).

Çevik yaklaşımların analiz süreçleri en basit haliyle KEŞFET – DOKÜMANTE ET -DOĞRULA -GELİŞTİR akışıyla ifade edilebilmektedir (Alexander & Beus-Dukic, 2009).

3.1.3. Geleneksel ve Çevik Analiz Süreçlerinin Önerilecek Yeni Model ve Süreç Açısından Değerlendirilmesi

Tez çalışmasının konusu olan ön-uç iş uygulamaları, kurumların web siteleri, pazarlama ve tanıtım portalleri, müşterilerine hizmet veren arayüzleri ya da çalışanlarının aktif olarak kullandıkları ortamlar olduğundan günümüz rekabet koşullarında sürekli değişen ve gelişen iş süreçlerine hizmet vermek durumundadır. İş kanunlarında, müşteri taleplerinde, rekabet ortamında hem gereksinimler hem de öncelikler çok hızlı değiştiğinden bu tip projelerde bir kapsam belirleyip o kapsamı sabitleyerek yazılım üretmek çok mümkün görünmemektedir. Bu nedenle tez çalışmasının konusu olan ön-uç uygulamalar için geliştirilecek olan yeni belirtim modelinin, çevik bir analiz süreciyle ortaya konulmasının doğru olacağı düşünülmüştür.

3.2. Artı ve Eksi Yönleriyle Gereksinim Belirtim Yaklaşımları

Tezin bu bölümünde çalışmanın ana öğesi olan gereksinim belirtim modelleri değerlendirilmekte ve farklı modeller birbirleriyle karşılaştırılmaktadır.

3.2.1. Metinsel Belirtim Yaklaşımları

Geleneksel belirtim yaklaşımlarının en başında IEEE'nin (1998) 830 nolu standardı gelmektedir. Bu standartta gereksinimler bir bütün olarak nasıl dokümante edileceğinin yanı sıra ayrıca her bir gereksinimin ve gereksinim olmayan ama ürünün ortaya çıkması için yapılacak projede ihtiyaç duyulacak diğer bilgilerin (varsayım, kısıt, risk) nasıl belirtileceği ve dokümante edileceği tarif edilmektedir. Standard, genel olarak

gereksinim ve ilişkili bileşenlerin metin olarak ifade edilmesine dayanmaktadır. Buna ek olarak gereksinimlerin metinsel belirtimi haricinde ekran görüntülerinin prototip olarak çizilmesi gerektiğini de ifade etmektedir.

IEEE'nin 830 numaralı standardı için verilebilecek birkaç örnek şöyledir:

- Sistem parola yenilenme esnasında girilen parolanın önceki 3 paroladan farklı olup olmadığın kontrol etmelidir,
- Kullanıcı parolasını unuttuğunda sistem kayıtlı eposta adresine parola sıfırlama linkini göndermelidir,
- Kullanıcı ATM'ye geçerli bir kartı ve pin kodu girerek mevcut hesabından günlük limit sınırları dahilinde para çekebilmelidir.

Bu modelin tespit edilen en temel eksikleri ise şöyledir:

- Bu model ön-uç uygulamalardaki kullanıcı-sistem etkileşimini iyi tanımlayamaz,
- Gereksinimleri belirli bir bağlam içinde algılamak zordur.

Modelin olumlu yanıysa düz bir metinsel biçimi oluşu nedeniyle eğitim ya da tecrübe gerektirmeksizin kullanılabilmesidir.

3.2.2. Formal Belirtimler

Formal modelleme dillerinin temel amacı, gereksinimleri belirli standartlara bağlayan bir ifade biçimi geliştirerek hem insanların hem de bilgisayar yazılımlarının anlayabileceği yapılar oluşturmaktır. Ek olarak bu yapıları kullanarak otomatik bir şekilde kodlar da üretilebilmektedir. Formal model tabanlı belirtim dillerinden en yaygın kullanılanları Process Specification Language (PSL), Structured Analysis and Design Technique (SADT), Scenario and Model Management (SAMM), Higher Order Software (HOS) ve Requirements Specification Language (RSL) olarak geçmektedir. Bunlar haricinde çok yaygın kullanılmasa da literatürde sayılan diğer modelleme dillerinden bazıları şöyle listelenebilir (Tse & Pong, 1991):

1. Clear and OBJ3,
2. ACT ONE, ACT TWO,

3. Larch,
4. SPECTRUM,
5. CASL.

PSL ile yazılmış bir belirtim örneği Şekil 3.2.2.1. de gösterilmektedir.

```
PROCESS:  process-order
        /* authors T.H. Tse and L. Pong */
    DESCRIPTION:
        this process captures the details of a valid order, calculates the
        amounts, discount and net-amount, and prepares an invoice;
    GENERATES:  net-invoice;
    RECEIVES:  valid-order, discount-rate;
    SUBPARTS ARE:  prepare-gross, compute-discount;
    PART OF:  process-sales;
    DERIVES:  amount
        USING:  price, quantity-ordered;
    DERIVES:  total-amount
        USING:  amount;
    DERIVES:  net-amount
        USING:  total-amount, discount-rate;
    PROCEDURE:
        1.  multiply price and quantity-ordered to obtain amount;
        2.  update stock record accordingly;
        3.  add amounts to obtain total-amount;
        4.  multiply total-amount by discount-rate to obtain net-amount;
        5.  update customer record accordingly;
        6.  generate invoice for net-amount;
    HAPPENS:  1 TIMES-PER valid-order;
    TRIGGERED BY:  valid-order-event;
    TERMINATION CAUSES:  invoice-event;
    SECURITY IS:  account-clerk-only;
```

Şekil 3.2.2.1. PSL ile Yazılmış Bir Belirtim Örneği (Tse & Pong, 1991)

3.2.3. Modern ve Çevik Gereksinim Belirtimi

Geleneksel olmayan gereksinim mühendisliğiyse zaman çerçeveli yaklaşım, yinelemeli gereksinim mühendisliği, paralel gereksinim geliştirme ve artırımlı geliştirme gibi başlıklar altında da incelenebilmektedir. Tüm bu yaklaşımlarda temel amaç hızla değişen iş dünyası gereksinimlerinin ve önceliklerinin en doğru şekilde anlaşılması ve çözümlenmesidir (Firesmith, 2003).

Çevik gereksinim belirtimi olarak belirli bir belirtim yöntemi resmi olarak tanımlanmasa da literatürde ve uygulamalarda büyük çoğunlukta kullanıcı hikayelerinin tercih edildiğini görmekteyiz (Bose & Kurhekar, 2014). Bunun haricinde çevik yöntemlerle sıklıkla olmasa da tercih edilen bir diğer yöntem de kullanım vakaları olarak

karşımıza çıkar. Kullanım vakaları aynı zamanda geleneksel yöntemde de kullanılmaktadır (Leffingwell & Widrig, 2003).

Kullanıcı hikayeleri detaylı analiz belirtiminden daha çok yazılımcıyla ürün sahibi arasındaki iletişim için bir dayanak ya da referans noktası olmaktadır. Bir kullanıcı hikayesi temel olarak kart, konuşulan önemli noktalar ve teyit bölümlerinden oluşur. "Kart" isimlendirmesi, kullanıcı hikayeleri ilk defa kullanıldığında kartondan yapılmış kartlar kullanıldığı içindir. Bu bölüm kartın ön yüzünü ifade eder ve belirli bir kullanıcının sistemden hangi gerekçeyle hangi işi ya da işlemi istediği belirtilir. İkinci bölümünde ürün sahibiyle yazılımcı ya da gereksinimleri aldığı paydaş arasında geçen görüşmeden notlar bulunur. Bu notlar temel iş kuralları, akış, genel hatlarıyla ekran prototipi, vb. olabilmektedir. Teyit bölümündeyse kabul kriteri olarak bilinen başarı sınama koşullar yazılmaktadır (Patton, 2014; Rubin, 2013).

Kullanıcı hikayesi için-birkaç örnek şöyle verilebilir:

- Kayıtlı bir kullanıcı olarak ürün araması yapabilmeliyim, böylece istediğim ürünü daha kolay bulabilirim,
- İçerik yöneticisi olarak kataloğa yeni ürünler ekleyebilmeliyim, böylece yeni ürünleri satışa sunmuş olurum,
- Banka müşterisi olarak hesabımdan para çekebilmek istiyorum, böylece gerektiğinde kolaylıkla paraya ulaşabilirim.

Bu modelin tespit edilen eksikler şöyle sıralanabilir:

- Bu model ön-uç uygulamalardaki kullanıcı-sistem etkileşimini iyi tanımlayamaz,
- Gereksinimleri belirli bir bağlam içinde algılamak zordur,
- Gereksinimlerin detayları yoktur. Dokümantasyona yönelik değil, daha çok iletişim için hatırlatıcı bir rol oynar.

Kullanıcı hikayelerinin olumlu yönleri şöyle sıralanabilir:

- Kapsamlı bir dokümantasyona dayanmadığından çevik süreçlere uygun bir şekilde çok kısa sürelerde oluşturulabilirler,

- Yüz yüze iletişimle desteklendiklerinden vakit kazandırarak doğru ürünün geliştirilmesine fayda sağlarlar,
- Modeli kullanmak için kapsamlı bir eğitim almaya veya tecrübeye gerek duyulmaz.

Diğer bir model olan kullanım vakaları, bir aktörün sistemle etkileşimini temsil ederler. Her bir kullanım vakası aktörün sistemde yerine getirmek istediği bir amacı temsil eder. Aktörse sistemdeki temel kullanıcı gruplarını ve kullanım vakalarını çalıştıran diğer sistemler ve zamanlayıcılar gibi unsurları temsil eder. Kullanım vakaları, kullanım vaka diyagramı ve kullanım vaka detayı olarak iki temel bileşenden oluşmaktadır. Hedeflenen yazılım "sistem" olarak isimlendirilir. Aktörlerle bu sistem içindeki kullanım vakaları arasındaki ilişkiyi gösteren diyagrama kullanım vaka diyagramı denir. Kullanım detayı ise her bir kullanım vakası için tetikleyicinin, birincil aktörün, temel ve ikincil adımlar gibi detay öğelerin neler olduğunun açıklandığı bölümdür (Leffingwell & Widrig, 2003; Metz, O'Brien, & Weber, 2003; Kulak, & Guiney, 2003).

Kullanım senaryosu detayı için örnek bir şablon aşağıdaki başlıklardan oluşmaktadır:

- Kullanım vakası numarası,
- Kullanım vakası adı,
- Oluşturan,
- Oluşturma tarihi,
- Son güncelleyen,
- Son güncelleme tarihi,
- Birincil aktör,
- İkincil aktörler,
- Açıklama,
- Kullanım vakası başlatıcı(lar),
- Ön koşullar,

- Ana akış,
- Alternatif akışlar,
- Başarı kriterleri,
- Çıkış durumu,
- Bağlı kullanım vakaları,
- Özel gereksinimler,
- Varsayımlar,
- Açık konular,
- İş kuralları,
- Test ve kabul sorumlusu.

Kullanım senaryosu için verilebilecek örnek bir senaryo ATM'den para çekmek olarak gösterilebilir. Bu senaryonun adımları şu şekilde olacaktır:

- Aktör geçerli bir kart takar – sistem pin giriş ekranını görüntüler,
- Aktör geçerli bir şifre girer – sistem şifrenin doğruluğunu onaylar,
- Sistem ana menü ekranını yükler,
- Aktör para çekme seçeneğini çalıştır – Sistem para çekme ekranını yükler,
- Aktör para çekeceği hesabı seçer ve miktarı girerek işlemi onaylar – Sistem müşteri bakiyesinin yeterliliğini onaylar,
- Sistem günlük para çekme limitinin uygunluğu onaylar,
- Sistem geçerli kupür ve miktarda paranın ATM de olduğunu onaylar,
- Sistem lütfen kartınızı alın mesajı olan ekranı gösterir ve kartı kullanıcıya verir,
- Aktör kartı alır – sistem parayı verir ve parayı alın mesaj ekranını görüntüler,
- Aktör parayı alır – sistem çıkış yaparak müşteri karşılama ekranını görüntüler.

Yukarıdaki ana başarı senaryosuna ek olarak aktörü başarılı bir işlem tamamlamaya ulaştırabilecek alternatif senaryolar şunlar olabilir:

- Kart geçersiz senaryosu,
- Kart çalıntı senaryosu,
- Pin kodu yanlış senaryosu,
- Hesapta yeterli para yok senaryosu,
- Günlük limit aşıldı senaryosu,
- Kasada yeterli para yok senaryosu,
- Aktör işlemler arasında 45 saniyeden fazla bekledi senaryosu.

Bu modelin tespit edilen eksikleri:

- Basit parametre yönetimi, ekle, sil, güncelle, listele gibi işlemler için karmaşıktır,
- Ön koşulların eksik olduğu durumlar yönetilmemektedir,
- Zamansal kesintiler ve aktör değişimlerini detaylandırmaz,
- Modellenmesi ve dokümantasyonunun karmaşıklığı nedeniyle eğitim ve deneyim gerektirmektedir.

Modelin olumlu yönleriyse şu şekilde sıralanabilir:

- Kapsamlı dokümantasyonu sayesinde analiste bağımlılıktan kurtulur ve kurumsal bilgi birikimi oluşumuna katkıda bulunur,
- Yüksek seviyeli diyagramı sayesinde problem ve çözüm alanlarının bütünsel olarak görülmesini sağlar,
- Kullanıcı ile sistem etkileşiminin yüksek olduğu, sürece dayalı uygulamaları geliştirmek için idealdir.

3.3. Belirtim Yaklaşımlarının Birbirleriyle Karşılaştırılması

Miller (1956) tarafından yapılan çalışmanın sonucu insan beyninin bir kerede ancak 5 ila 9 arasındaki öğeyi hatırlayabildiği ve işleyebildiğini göstermektedir. Bunun sonucu olarak özellikle geleneksel yöntemlerin kullandığı "sistem …. yapmalı" şeklindeki onlarca ya da yüzlerce gereksinimden oluşan belirtimler hem kavranması, hem anlaşılması hem de yönetilmesi zor bir belirtim biçimi olarak karşımızda durmaktadır. İnsan beyni yazılı metinlerden çok görsel ifadelere, çizimlere, şekillere karşı daha duyarlıdır ve bu şekildeki ifadeleri daha hızlı ve doğru değerlendirir (Beatty & Chen, 2012). Bu nedenle metinsel belirtimlerdense modeller, diyagramlar ya da görseller kullanmak daha etkili olacaktır.

Formal modeller, iş uygulamaları alanında kullanımına rastlanmadığı için değerlendirmeye alınmamıştır. Ancak formal modeller de geleneksel yöntemlere benzer şekilde metinsel ifadelerden oluştuklarından ve hatta yapıları gereği daha çok programlama betiklerine benzediklerinden aynı çerçevede değerlendirilebilirler.

Hem geleneksel hem de çevik yöntemlerde kullanılan kullanım vakaları, kullanım vaka diyagramlarıyla aslında metinsel ifadenin ötesine geçip modellemeyi kullansa da bazı açılardan eksik kalmaktadır.

Bu eksiklerden en kritik olanları şu şekilde sıralanması mümkündür (Firesmith, 2007; Lilly, 1999):

- Kullanım vakalarının işlevsel gereksinimlere odaklı olması ve işlevsel olmayan gereksinimlerin belirtiminde zayıf kalması,
- Bir aktörün çok sayıda kullanım vakasıyla ilişkili olması ya da bir kullanım vakasının çok sayıda aktör tarafından kullanılması durumunda diyagramın okunamaz hale gelmesi,
- Çok küçük gereksinimlerle çok büyük gereksinimlerin aynı yapıyla ifade etmenin zorluğu (çok farklı büyüklük ve karmaşıklıkta gereksinimlerin aynı yapıyla ifade edilememesi),
- CRUD (create – retrieve – update –delete) ile tanımlanan, verilerin temel kayıt, güncelleme, silme ve getirme/listeleme gibi işlemleri için uygun olmaması,

- Kullanıcı arayüzleri ve etkileşimiyle ilgili herhangi bir bilgi içermemesi,
- Kullanım vaka detayının ana, alternatif ve istisna senaryolarının karmaşıklaşabilmesi.

Diğer taraftan çevik yöntemlerde dokümantasyonun amacı konuşulan şeylerin hatırlanmasına yöneliktir bir referans noktası oluşturmaktır. Kurumsal bilgi birikimi oluşturmak ya da bir bilgi aktarımını tüm detayıyla aktarmak hedeflenmemektedir. Bu nedenle kullanıcı hikayeleri doğası gereği bir belirtim ve dokümantasyon olarak sadece çok genel bir başlık olarak yer almaktadır (Patton, 2014).

Tüm bu detaylara bakıldığında geleneksel yöntemlere yönelik metotlarda gerekli bilgilerin çoğunluğu yer almakla birlikte bilgilerin ve dokümantasyonun organizasyonu, anlaşılması ve kavranması çok zor olmaktadır. Diğer taraftan kullanım vakaları daha görsel ve çeşitli modeller ve ekran prototipleriyle desteklendiğinde daha anlaşılabilir olmakla birlikte, bu tip bir uygulama için tarif edilmiş bir süreç ya da çerçeve bulunmamaktadır. Aynı zamanda kullanım vakalarının yukarıda tanımlanan eksikleri tek başına kullanılmasının zorluğunu ortaya koymaktadır. Kullanıcı hikayeleri ise daha çok yüz yüze iletişim için hatırlatıcı olarak kullanıldığından diğer yaklaşımlara göre çok zayıf kalmaktadır.

BÖLÜM 4

YENİ MODEL ÖNERİSİ

Bu bölümde çalışmanın ana amacı olan gereksinim belirtim modeli ortaya koyulmaktadır. Model detaylarıyla anlatılmadan önce model geliştirme yöntemleri ve standartları ele alınmış, ortaya konulacak olan model için gereksinimler tespit edilmiştir. Ayrıca mevcut yöntemlerin eksikleri ve önerilen sistemde hedeflenen özellikler de detaylarıyla anlatılmıştır. Ardından model temel bileşenleri, bileşenlerin birbiriyle ilişkileri, analizin nasıl bir süreçle gerçekleştirileceği, belirtimin nasıl yapılacağı, emek tahminleme yöntemi ve hangi adımlarda hangi çıktıların oluşturulacağı tarif edilmiştir. Bölümün sonunda modelin analistler tarafından pratik bir şekilde kullanılabilmesi ve analiz hatalarının minimuma indirilmesi amacıyla çalışmanın bir parçası olarak geliştirilmiş yazılım açıklanmaktadır.

4.1. Önerilecek Model İçin Yapılan Hazırlıklar

Yeni model ortaya konmadan önce yapılan çalışmalarda ilk olarak yeni modelin sahip olması gereken nitelikler ve modelin çıktısının neler olması gerektiğine çalışılmıştır. Ardından yazılım gereksinim belirtimi ve yönetimi üzerine standartlar incelenerek belirlenen gereksinimlerin nasıl yapılandırılması gerektiği çalışılmıştır. Bu aşamada son olarak model ve meta model kavramları ve nasıl tasarlanacakları incelenerek, ortaya konulacak yeni model için gerekli hazırlıklar tamamlanmıştır.

4.1.1. Mevcut Gereksinim Modellerindeki Eksikler ve Ortaya Konulması Hedeflenen Modelin Gereksinimleri ile Potansiyel Çıktıları

Önceki bölümlerde ortaya koyulan bilgiler ve tespitler doğrultusunda öncelikle mevcut gereksinim analizi model ve süreçlerindeki eksikler şu şekilde belirlenmiştir:

1. Kullanıcı hikayesi yaklaşımı genel olarak uygulanması kolay, analiz çıktılarının üretimi hızlı bir yaklaşımdır. Ancak ortaya koyulan çıktılar bir gereksinim dokümanından çok yapılacak işle ilgili fikir verebilecek düzeyde zayıf bilgilerden oluşur. Tüm iş kurallarını, işlevsel ve işlevsel olmayan gereksinimleri içermeyen, kullanıcı etkileşimi ve veri yapısını tarif etmeyen bir çıktı üretir.
2. Kullanım vakaları yaklaşımı işlevsel gereksinimleri detaylı olarak ele almakla beraber işlevsel olmayan gereksinimler, kullanıcı etkileşimi ve veri yapısının nasıl ele alınacağına dair bir yönlendirmede bulunmamaktadır. Ayrıca akış süreçlerinden oluşmayan basit işlevlere sahip yazılımlar için fazla karmaşık bir yapıdır.
3. SRS ve diğer standart dokümanlara yönelik çalışmalarda bunların başlıklarının ve yapılarının ortaya koyulduğu ancak bu başlıkların altlarının nasıl doldurulacağının tarif edilmediği görülmüştür.

Literatür taraması, yazılım gereksinimleri ve yazılım gereksinim analiziyle ilgili tanımların değerlendirilmesi, mevcut belirtim modelleri ve analiz süreçlerinin incelenmesi ve karşılaştırılması sonucunda ortaya konacak yeni analiz modeli için aşağıdaki kriterlerin ele alınmasının gerekliliği tespit edilmiştir.

1. Model, kullanıcı ve sistem gereksinimleri bir arada ve birbirleriyle ilişkilerini de ortaya koyarak tarif etmelidir.
2. Model hem yeni geliştirilecek projelerde hem de mevcut yazılımlarla ilgili değişiklik taleplerinde kullanılabiliyor olmalıdır.
3. Model, geliştirilecek olan yazılımın diğer yazılımlarla olan entegrasyonunu en uygun şekilde tarif edebiliyor olmalıdır.

4. Model, yazılım projesinin teknik olan (yazılım mimarı, yazılım uzmanı, test uzmanı) ve olmayan (iş birimi alan uzmanları, anahtar kullanıcılar, sponsor) tüm paydaşları tarafından kolaylıkla okunabilmeli ve anlaşılabilmelidir.
5. Model, işlevsel olmayan gereksinimler olarak da bilinen ve çoğunlukla yazılımın kalite özellikleri olarak nitelendirilen, güvenlik, kullanışlılık, performans gibi kısıtları da belirtebilmelidir.
6. Gereksinimlerin edinilmesi, analiz sürecinin doğası gereği aşamalı detaylandırılmayla mümkündür. Modelin süreci destekleyebilmesi için üst seviyede kapsam tarifini de içeriyor olması, büyük resimden başlayarak kademeli olarak detaylandırılabilecek bir seviyelendirmeye sahip olması gerekmektedir.
7. Analiz modeli birbiriyle ilişkili olan çok sayıda detaylı gereksinimi anlamlı bir şekilde ve kullanıcıya değer katacak bileşenler olarak gruplandırabilmelidir. Böylece yazılımın kullanıcıya üreteceği değeri ortaya koyacak soyut bir kavram üzerinden gereksinimler tarif edilebilecektir. Aynı zamanda özellik seviyeleri çerçevesinden soyutlama yapılarak sistem farklı seviyelerde tarif edilebilecektir.
8. Model, kullanıcı ve sistem arasındaki etkileşimi tarif etmeli, böylece kullanıcının hangi aksiyonları aldığında sistemin hangi cevapları vereceği ve sistemde oluşan bazı durumların kullanıcıya nasıl aktarılacağı net bir biçimde ortaya koyulabilmelidir.
9. Model, kullanıcının sistem üzerinde gerçekleştirmek istediği iş ve işlemlerle ilgili olası tüm senaryoları kapsayacak ve senaryoların birbirleriyle ilişkisini de gösterecek bir yapıya sahip olmalıdır.
10. Model iş süreçlerindeki kritik iş kurallarından biri olan iş akışlarını tarif edebiliyor olmalıdır. Ayrıca birden fazla yazılım, birden fazla kullanıcı oturumu, birden fazla ekran üzerinden geçen iş akışlarını da yönetebilmelidir.
11. Model, ekran görüntülerini ve ekran görüntülerinin diğer model öğeleriyle ilişkilerini tarif etmeli, böylece kullanıcı gereksinimlerinin tüm paydaşlar tarafından en uygun şekilde anlaşılmasına olanak sağlamalıdır.

12. Model, kullanıcılarla etkileşimde verilecek bilgiler, mesajlar ve bildirimleri belirtebiliyor olmalıdır. Bu etkileşimde çevrimiçi olmayan durumlar da göz önünde bulundurulmalıdır. (Örneğin e-posta gönderimi.)
13. Model, kullanıcı tarafında iş kurallarının en uygun şekilde işletilebilmesi için kullanıcı girişlerinin doğrulamalarını net bir biçimde tarif edebilmelidir.
14. Model yazılımda veri yönetiminde kullanılacak varlıkları, bu varlıkların birbirleriyle ilişkilerini ve niteliklerinin yapısını, verinin nasıl saklanacağını ve verinin statik olarak hangi kısıtlarla, iş kurallarıyla çalışacağını ortaya koymalıdır.
15. Model, tarif edilen ihtiyaçlar ve belirlenen gereksinimler doğrultusunda ortaya konan problemin büyüklüğünü ve karmaşıklığını ortaya koyabilen bir ölçümleme tarif edebilmelidir.
16. Model görünümler ve aksiyonlar perspektifinde rol tabanlı yetkilendirmeyi belirtiyor olmalıdır.
17. Model, sistem içerisinde gerekli olan yerlerde durum ve geçiş tarifini yapabilmelidir.
18. Model, fiziksel tasarıma girmeden, problemin tüm boyutlarıyla büyük resim çerçevesinden de değerlendirilebilmesi için kavramsal tasarım seviyesinde bir modelleme de sunmalıdır. Bu kavramsal model içerisinde çözümün temel bileşenleri olan akışlar, iş kuralları, ekranlar, entegrasyonlar, özellikler, işlevler, veri modeli tarif edilebilmelidir.
19. Model, çözümün gerekli olan tüm yönleriyle belirtiminin yapabiliyor ve analizdeki her bir öğenin nasıl, ne zaman yapılacağını ve birbirleriyle olan ilişkisini tarif edebiliyor olmalıdır.
20. Model, eksiksiz ve herkesçe aynı şekilde uygulanabilir olmalı ancak gerektiğinde projeye ve kuruma uygun bir şekilde esnetilebilmelidir. Bunun için model aynı zamanda belirtim yapısıyla birlikte analiz sürecini de tarif etmelidir.

Modelle ilgili tüm bu maddeler gerçekleştiğinde potansiyel olarak üç ana çıktı tarif edilebilir:

- Yazılım gereksinim yönetimi, gereksinimlerin toplanması, analizi, modellenmesi, belirtimi, doğrulanması, dokümantasyonu sürecinin tarif edilmesi,
- Bu süreç boyunca gereksinimlerin depolanması ve dokümantasyonunda kullanılacak olan belirtim modelinin ortaya konması,
- Analiz sonucunda ortaya konulacak analiz çıktıları üzerinden çözümün karmaşıklık ve büyüklüğüyle ilgili bir tahminleme yönteminin ortaya konması

4.1.2. Yeni Modelin Kurgulanması İçin İncelenen ve Faydalanılan Standartlar

Çalışmanın bu aşamasında, yeni bir model ortaya koymadan önce, mevcut modellerin dayandığı ve yeni bir model oluşturulurken mutlaka göz önünde bulundurulması gereken uluslararası standartlar incelenmiştir. Bu standartları belirlerken esas alınan en temel kaynak "IEEE Yazılım ve Sistem Mühendisliği Standartları Temel Koleksiyon" isimli yayındır (https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tocs/softess.pdf). Yayında belirtilen en temel standartlar şunlardır:

- 730-2002 Yazılım Kalite Güvence Planları İçin IEEE Standardı,
- 828-2005 Yazılım Konfigürasyon Yönetim Planları İçin IEEE Standardı,
- 830-1998 Yazılım Gereksinim Belirtimi İçin Tavsiye Edilen Uygulama,
- 1008-1987 (R2002) Yazılım Birim Testi İçin Standart,
- 1012-2004 Yazılım Doğrulama ve Geçerleme İçin IEEE Standardı,
- 1016-1998 Yazılım Tasarım Açıklamaları İçin Tavsiye Edilen Uygulama,
- 1028-1997 Yazılım Gözden Geçirmeleri İçin IEEE Standardı,
- 1058-1998 Yazılım Yönetim Planları İçin IEEE Standardı,
- 1063-2001(R2007) Yazılım Kullanıcı Dokümantasyonu İçin IEEE Standardı,

- 1074-2006 Yazılım Geliştirme Projesi Yaşam Döngüsü Süreci İçin IEEE Standardı,
- 12207:2008 Sistem ve Yazılım Mühendisliği – Yazılım Yaşam Döngüsü Süreci,
- 14764:2006 Yazılım Mühendisliği İçin Standart – Yazılım Yaşam Döngüsü Süreci- Bakım,
- SWEBOK Kılavuzu.

Yeni bir gereksinim belirtimi için göz önünde bulundurulması gereken standartları en iyi özetleyen kaynaklardan biri de Schmidt (2000) in "Implementing the IEEE Software Engineering Standards" kitabıdır. Kitapta yazılım mühendisliği alanındaki standartlar ele alınmış ve uygulamaya yönelik tavsiyelerde bulunulmuştur.

Kitapta yazılım geliştirme süreçlerinin iyileştirilmesine yönelik izlenilmesi önerilen minimum standart listesi şu şekilde verilmektedir:

- IEEE Std. 730: Yazılım Kalite Güvence Planları,
- IEEE Std. 828: Yazılım Konfigürasyon Yönetimi Planları,
- IEEE Std. 830: Yazılım Gereksinim Belirtimi,
- IEEE Std. 1012: Yazılım Doğrulama ve Geçerleme,
- IEEE Std. 1016: Yazılım Tasarım Açıklamaları,
- IEEE Std. 1063: Yazılım Kullanıcı Dokümantasyonu.

Schmidt (2000) bu standartlar incelendikten sonraki çalışmalar için de aşağıdaki listeyi önermektedir:

- IEEE Std. 829: Yazılım Test Dokümantasyonu,
- IEEE Std. 1008: Yazılım Birim Testi,
- IEEE Std. 1028: Yazılım Gözden Geçirmeleri,
- IEEE Std. 1058: Yazılım Proje Yönetim Planları,
- IEEE Std. 1228: Yazılım Güvenlik Planları.

Hiç şüphe yok ki bu standartlar içinde çalışmanın amacıyla en yakından ilişkili ve mutlaka çalışmaya dahil edilmesi gereken standart 830 numaralı "Software Requirements Specifications"dır. Literatür taraması esnasında bu standardın ISO / IEC / IEEE 29148: 2011 olarak yenilendiği tespit edilmiştir. Standartta iyi bir yazılım gereksinimleri spesifikasyonunun (SRS) içeriği ve nitelikleri açıklanmış ve birkaç örnek SRS ana hatları sunulmuştur. Bu önerilen uygulama, geliştirilecek yazılımın gerekliliklerini belirlemeyi amaçlar, ancak kurum içi ve ticari yazılım ürünlerinin seçimine yardımcı olmak için de uygulanabilir.

Çalışmada en çok atıfta bulunulan ve yararlanılan kaynaklardan biri olan SWEBOK kılavuzu modeli yapılandırırken başvurulan standartlardan ikincisi olmuştur. Bu kılavuz yazılım mühendisliğini, gereksinim mühendisliği de dahil olmak üzere tüm hatlarıyla uçtan uca anlatmaktadır.

Yeni bir modeli oluşturmayla ilgili esasları belirlemek ve bu konuda izlenecek yolu belirlemek için 1074-2006 numaralı "Yazılım Geliştirme Projesi Yaşam Döngüsü Süreci İçin IEEE Standardı, Sistem ve Yazılım Mühendisliği – Yazılım Yaşam Döngüsü Süreci" ve 14764:2006 numaralı "Yazılım Mühendisliği İçin Standart – Yazılım Yaşam Döngüsü Süreci – Bakım" standartları incelenmiştir.

Birçoğu yazılım ürününün ve sürecinin doğruluğuna yönelik olmasına rağmen, gereksinim yönetimi sürecine de doğrulama ve geçerleme görevleriyle yön verebilecekleri düşüncesiyle 730 numaralı "Yazılım Kalite Güvence Planları", 1012 numaralı "Yazılım Doğrulama ve Geçerleme", 829 numaralı "Yazılım Test Dokümantasyonu" 1008 numaralı "Yazılım Birim Testi" ve 1028 numaralı "Yazılım Gözden Geçirmeleri" standartları incelenmiştir.

Çalışmanın başında belirlenen listede yer alan aşağıdaki standartlar, çalışmanın kapsamının dışında oldukları için değerlendirmeye alınmamıştır.

- IEEE Std. 828: Yazlım Konfigürasyon Yönetimi Planları,
- IEEE Std. 1016: Yazılım Tasarım Açıklamaları,
- IEEE Std. 1063: Yazılım Kullanıcı Dokümantasyonu,
- IEEE Std. 1058: Yazılım Proje Yönetim Planları,

- IEEE Std. 1228: Yazılım Güvenlik Planları.

4.2. Model ve Meta Model

Haggett ve Chorkey modelleme için literatürde yapılan tanımlardan üzerinde en çok anlaşılan ve tutarlı bulunanı, "bir modelin gerçeğin basit bir şekilde soyutlanması" olduğunu ifade etmektedir (Haggett & Chorkey, 1967). Selic (2003) e göre bir model faydalı ve yararlı olmanın dışında aşağıdaki özellikleri sergiliyor olmalıdır:

1. Soyutlama: bir model her zaman temsil ettiği sistemin indirgenmiş bir ifadesi, soyutlamış bir hali olmalıdır. Model orijinalden daha detaylıysa model olarak ifade edilemez.
2. Anlaşılabilirlik: Bir modelin var olan bir sistemi temsil ediyor olması yetmez aynı zamanda anlaşılabilir de olmalıdır.
3. Doğruluk: Bir model gerçeğinin doğru bir temsili olmalıdır.
4. Tahmin edilebilirlik: Modele bakıldığında bütünle ilgili her şey detaylıca ortaya konmamış bile olsa, bütünün nasıl olduğuyla ilgili bir fikir verebiliyor olmalıdır.
5. Düşük maliyet: Bir sistemin modellenmesi anlamlı bir bütçe ile gerçekleştirilebiliyor olmalıdır.

Model, gerçek yaşamdaki bir şeyin soyutlanarak ifade edilmesiyken, meta-model ise modelin soyutlanarak ifade ve temsil edilmesi anlamına gelmektedir (Harel & Rumpe, 2000). Kısaca meta-model, "modelin" modelidir.

4.3. Önerilen Modelin Temel Bileşenleri ve Genel Bakış

Bu başlık altında analiz belirtim modelini oluşturan temel öğeler, bu öğeler birbirleriyle ilişkileri ve modelin meta-model olarak gösterimi ele alınacaktır.

Yeni model belirlenirken "Ortaya Konulması Hedeflenen Modelin Gereksinimleri ve Potansiyel Çıktıları" başlığı altında açıklanmış olan gereksinimler en başlıca faktör olarak ele alınmıştır. Bununla beraber yazılım gereksinim analizi, yazılım mimari

modelleme ve yazılım geliştirmede kullanılan çeşitli yaklaşımlar da incelenerek sentez bir model ortaya konulmuştur.

Öncelikle modelin belirtiminin temeline hem "özellik" hem de "senaryo" yaklaşımlarının sentezlenerek alınması kararlaştırılmıştır. Böylece bir yandan sistem bir özellikler hiyerarşisi yapısı içerisinde tanımlanabilecek ve büyük resim farklı seviyelerde ortaya konabilecek diğer yandan da iş akışına dayalı iş kuralları doğru şekilde tanımlanabilecektir.

Yeni modelin çözmeye çalıştığı en önemli sorun, kullanım vakaları gibi senaryoya (iş akışına) dayalı sistemlerde, akış içermeyen temel işlerin (kullanıcı ekle, sil, çıkart, listele) ve iş kurallarının (talep onayla, reddet, iptal et) ifadesinin güçlüğü olarak belirlenmiştir. Ayrıca senaryo temelli yaklaşımlarda dış sistemlerden başlayan ve iç sistemde devam eden kullanım vakaları (senaryolar) arasından ilerleyen iş akışları açıkça tarif edilememektedir. Bu yeni modelle hem dış sistemlerden hem de sistem içinden başlayan akışlar gösterebilmektedir. Ayrıca sistem akış içerisinde yer almayan işlevsel öğeleri de anlaşılabilir bir biçimde ifade edilebilmektedir.

Model belirlenirken incelenen önemli kavramlardan biri de yazılım modellemede yer alan katmanlardır. Bu modeller en temel haliyle karşımıza kullanıcı arayüz katmanı, iş kuralları katmanı ve veri erişim katmanı olarak çıkmaktadır. Ön-uç iş uygulamalarının analizinde de karşımıza benzer şekilde kullanıcı arayüzleri, iş kuralları ve iş öğeleri çıkmaktadır. Yeni modelde bu yaklaşımdan yola çıkılarak arayüzlerin ve verinin modellenmesi özel olarak ele alınmıştır. Buna ek olarak iş akışları iş kurallarından ayrılarak kendi başlarına bağımsız birer öğe olarak ele alınmıştır. Model içerisinde iş kuralları modelin en küçük ve en temel birimi olan işlevsel öz birim içerisinde tanımlanmaktadır.

Günümüzde iş dünyasında mümkün olduğunca tüm süreçlerin yazılımlarla çözülmeye çalışıldığı bir gerçektir. Ancak bir kurumun mevcut ve gelişen tüm ihtiyaçlarını karşılayacak tek bir yazılımın geliştirilmesi pek mümkün olmamakta, kurumlar bu nedenle farklı yazılımlar kullanmakta ve bunlar arasında entegrasyon yoluna gitmektedir. Bu sebeple modelde önemsenen bir diğer konu da dış sistemler entegrasyon konusudur ve bunun için ayrı öğeler tanımlanmıştır.

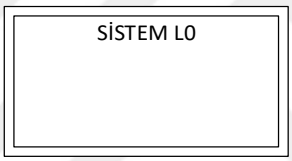
Bunların dışında bir gereklilik olarak işlevsel olmayan gereksinimler bağımsız bir öğe olarak ve tüm seviyelerde var olabilecek şekilde kurgulanmıştır. Böylece hem gösterimleri hem de kavranışları daha kolay hale getirilmeye çalışılmıştır.

Modeli oluşturan temel öğeler aşağıda başlıklar halinde sunulmaktadır.

4.3.1. Sistem

Sistem, ihtiyaçları karşılayacak olan yazılım ürününü, bunu kullanacak rolleri, işin gerçekleşmesi için gerekli olan akışları, alt sistemleri, arayüzleri, veri yapısını, işlevsel olmayan gereksinimleri ve işlevsel öz birimleri tümüyle kapsayan ana çerçevedir.

Sistem öğesi modelde Şekil 4.3.1.1. de görülen sembol ile temsil edilmektedir.



Şekil 4.3.1.1. Sistem Öğesinin Sembolü

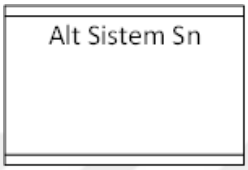
Her ne kadar çevik yöntemler ortaya çıkartılacak yazılımı bir ürün olarak görüp, "ürün" olarak isimlendirse de geleneksel yöntemlerde yazılım ürünü bir sistem olarak ifade edilmektir. Geleneksel yaklaşımdaki sistem kavramının, yukarıda tarif edilen çerçeve için daha uygun olduğu düşünülmektedir ve bu nedenle bu çalışmada sistem kavramı tercih edilmiştir.

Kendi başına bir çözüm olarak ortaya konulacak olan yazılım ürünleri birer sistem olarak düşünülebilir. Örneğin Avans Yönetimi Sistemi, Masraf Bildirimi Sistemi, İnsan Kaynakları Yönetimi gibi başlıklar altında geliştirilebilecek yazılımlar birer sistem olarak kabul edilebilir.

4.3.2. Alt Sistem

Bir sistem kendi içerisinde bir ya da daha fazla sayıda alt sistemden oluşabilmektedir. Alt sistemler literatürde ve sektörde kullanılan modül kavramı yerine de geçmektedir.

Alt sistem öğesi Şekil 4.3.2.1 de şekilde görülen sembol ile temsil edilmektedir.



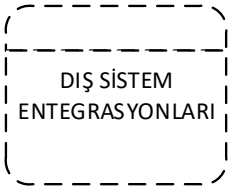
Şekil 4.3.2.1. Alt Sistem Öğesinin Sembolü

Sistemler alt sistemlere sahip olabildikleri gibi, ayrıca alt sistemler de kendi alt sistemlerini hiyerarşik bir yapı çerçevesinde içerebilirler. Bununla ilgili bir seviye sınırlandırması yapılmamıştır. Çünkü sistemin büyüklüğüne göre içi içe olan alt sistem hiyerarşisi de artabilecektir. Örneğin bir insan kaynakları yazılımda "seçme ve yerleştirme" işi bir alt sistemken, bir ERP yazılımı sistem olarak ele alındığında bunun altındaki insan kaynakları bir alt sistem, onun altındaki seçme ve yerleştirme de onun altındaki, üçüncü seviye alt sistem olarak karşımıza çıkacaktır.

4.3.3. Dış Sistem Entegrasyonları

Bir sistem bazen bir ya da daha fazla dış sistemle etkileşime girebilmektedir. Veri aktarımı, uzak prosedür kullanımı gibi amaçlar için dış sistemlere bağlanılması gerekebilir. Bunun için modelde "dış sistem entegrasyonu" bir öğe olarak yer almıştır.

Dış sistem öğesi modelde şekil 4.3.3.1.'de görülen sembol ile temsil edilmektedir.



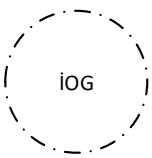
Şekil 4.3.3.1. Dış Sistem Öğesinin Sembolü

Dış sistem, sistem dışında başka bir sistem, servis, veri kaynağı ya da herhangi bir öğedir. Bizim sistemimizde önemli olan dış sistemle sistemimizin nasıl haberleştiği ve çalıştığıdır.

4.3.4. İşlevsel Olmayan Gereksinim

İşlevsel olmayan gereksinim, modelin en küçük öğesi olan işlevsel öz birimi tanımlayan bir karakteristik özellik olarak tanımlanabileceği gibi aynı zamanda her seviyedeki alt sistem ve sistemin tamamında geçerli bir kısıt da olabilmektedir. Örneğin "Personel İş Avansı Talebi ve Masraf Bildirim Yönetimi" sisteminin bir web uygulaması olarak tasarlanması gereksinimi bir işlevsel olmayan gereksinimdir ve sistemin tümünü etkilemektedir. Diğer taraftan "masraf bildirim arayüzü mobil cihazlardan da kolaylıkla kullanılabilmelidir" şeklindeki bir gereksinim yine işlevsel olmayan bir gereksinim olmakla beraber sadece "Avans Yönetimi" alt sistemi altındaki belirli bir işlevsel öz birimi etkilemektedir.

İşlevsel olmayan gereksinim (İOG) öğesi Şekil 4.3.4.1 de görülen sembol ile temsil edilmektedir.



Şekil 4.3.4.1. İOG Öğesinin Sembolü

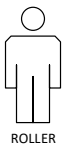
İOG için İÖB seviyesindeki bir örnek, ilgili İÖB'nin ekranının yüklenme zamanının maksimum 3 saniye olması şeklinde verilebilir.

4.3.5. Kullanıcı Rolleri

Sistemin var oluşu gerçek yaşamdaki insanlar için bir değer üretmektir. Sistemler ya başka sistemler üzerinden ya da direkt olarak kullanıcılar tarafından kullanılarak değer üretmektedirler. Bu nedenle sistemler kullanıcılar tarafından çalıştırılmaktadır.

Bir sistemde hangi rollerin olduğu, bu rollerin hangi genel gereksinimleri olduğu, bilgisayar ve yazılımlarla ilgili bilgi ve beceri seviyeleri etkili bir yazılım geliştirebilmek için önemli kavramlardır. Ayrıca rol bazlı yetkilendirmeler için rollerin belirlenmiş olması ve bu rollerin hangi operasyonları gerçekleştirebileceğinin tanımlanması gerekmektedir.

Kullanıcı rolü öğesi modelde Şekil 4.3.5.1. de görülen sembol ile temsil edilmektedir.



Şekil 4.3.5.1. Rol Öğesinin Sembolü

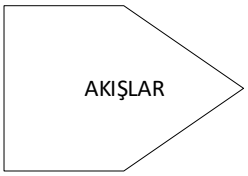
Modelde roller sistem seviyesinde tanımlanmıştır ve sistemin tüm öğeleriyle etkileşimde olabilmektedirler.

4.3.6. İş Akışları

İş akışları sistem seviyesinde tanımlanmış öğelerdir. İş akışlarının alt sistem içindeki yansımaları birincil ve ikincil senaryolar olarak tarif edilmektedir. Bir iş akışı sistem içinden başlatılabileceği gibi dış sistemler tarafından da başlatılabilir ve entegrasyon noktaları aracılığıyla sisteme geçerek buradan devam edebilmektedir. Aynı

şekilde sistem içinde ilerleyen bir iş akışı yine entegrasyon noktası aracılığıyla başka bir dış sisteme geçerek devam edebilmektedir.

İş akış öğesi modelde Şekil 4.3.6.1. de görülen sembol ile temsil edilmektedir.



Şekil 4.3.6.1. Akış Öğesinin Sembolü

4.3.7. Arayüzler ve Kullanıcı Etkileşimleri

Kullanıcıya bilgi gösterecek, kullanıcıdan bilgi alacak ve kullanıcıyla etkileşimi sağlayacak olan öğe "arayüz"dür. Arayüzler sistem seviyesinde tanımlanmaktadır. Arayüz en az bir işlevsel öz birimle ilişkilidir ve onun işlevini yerine getirmek üzere tanımlanmaktadır. Ancak bazen bir arayüz birden fazla işlevsel öz birimle ya da bir öz birim birden çok öz birimle ilişkili olabilmektedir. Örneğin bir portaldeki ana sayfada portalin çeşitli işlevlerinden özet bilgiler, bildirimler ve görüntüler gösterilebilmektedir.

Bunun dışında ön-uç uygulamalarda arayüzler ile birlikte bildirim ve mesaj gibi diğer etkileşim unsurları da bulunabilmektedir. Bunlar da dahil tüm kullanıcı etkileşimi kapsaması açısından bu öğenin adı kullanıcı etkileşimi olarak belirlenmiştir.

Kullanıcı etkileşimi öğesi Şekil 4.3.7.1. de görülen sembol ile temsil edilmektedir.

Şekil 4.3.7.1. Kullanıcı Etkileşimi Öğesinin Sembolü

4.3.8. Veri Yapısı

Veri yapısı, sistemde kullanılacak olan veri modelindeki her bir alanın/niteliğin (field/attribute) teker teker ele alınarak statik veri kısıtlarının ve kullanıcı veri girişi doğrulamalarının tanımlandığı öğedir.

Veri yapısı veri modelinin detay bileşenidir sistemin tüm seviyeleriyle ve işlevsel öz birimlerle ilişkilidir. İşlevsel öz birimlerde yazılım gereksinimleri yerine getirilirken kullanılan veri ve bilginin yapısını tarif etmektedir.

Veri yapısı öğesi Şekil 4.3.8.1. de görülen sembol ile temsil edilmektedir.

Şekil 4.3.8.1. Veri Yapısı Öğesinin Sembolü

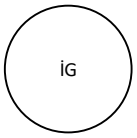
Bir avans bildirimi için veri yapısı avans talebinde bulunan kullanıcı, avans talep tarihi, avans miktarı, avans talep açıklaması gibi detaylardan oluşmaktadır.

Veri yapısı sistem seviyesinde tanımlanan bir öğedir.

4.3.9. İşlevsel Gereksinim

İşlevsel gereksinimler, modelin en küçük öğesi olan işlevsel öz birimin yerine getirmesi gereken ya da karşılaması gereken temel davranışsal gereksinimler olarak tanımlanabilirler. Formüller, hesaplamalar, kısıtlar gibi iş kuralları, sistemin uygulaması gereken karar unsurları ve yerine getirmesi gereken aksiyonlar işlevsel gereksinimlerin en önemli unsurlarıdır.

İşlevsel gereksinim (İG) öğesi Şekil 4.3.9.1 de görülen sembol ile temsil edilmektedir.



Şekil 4.3.9.1. İG Öğesinin Sembolü

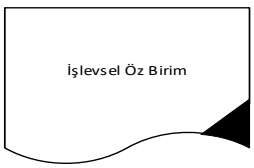
4.3.10. İşlevsel Öz Birim

İşlevsel öz birim (İÖB) yeni modelin temelini oluşturan en önemli öğedir. İÖB, kullanıcının sistemde yerine getirmeyi amaçladığı en atomik işlevi temsil eder. Bu işlev bir ya da birkaç işlevsel gereksinimi ve bununla birlikte varsa ilgili işlevsel olmayan gereksinimleri yerine getirir. İÖB bir sistemin ya da alt sistemin alt öğesidir ve diğer öğelerin çoğuyla ilişkilidir. Çizelge 4.3.10.1 de İÖB ile modeldeki diğer öğelerin ilişkisini göstermektedir.

Çizelge 4.3.10.1. İşlevsel Öz Birimin Diğer Öğelerle İlişki Yapısı

Model Öğe Adı	**İÖB ile İlişkisi**
Sistem / Alt Sistem	İÖB mutlaka bir sistem ya da alt sistemin alt öğesi olan modülün içinde yer alır.
Dış Sistem Entegrasyonu	İÖB amacını yerine getirirken bir dış sistemle bilgi alışverişinde bulunabilir. Örneğin "Bağlı Personelin İzin Taleplerini Listeleme" İÖB'si "Active Directory Entegrasyonu" ile birlikte çalışmak durumunda olabilir.
İşlevsel Olmayan Gereksinim	Bazı İÖB'ler kullanıcı tarafından belirtilen işlevsel olmayan gereksinimleri yerine getirmek durumunda olabilir.
Kullanıcı Rolleri	Bir İÖB mutlaka en az bir rolle ilişkili olmak durumundadır. İÖB'yi roller çalıştırır. Bir İÖB birden fazla rol tarafından kullanılabilir.
İş Akışları	İÖB bir iş akışının bir ya da daha fazla adımının alt parçası olabilir.
Kullanıcı Etkileşimi	Kullanıcı etkileşimi olan İÖB'ler bir ya da daha fazla arayüz, mesaj ve/veya bildirim kullanabilir.
Veri Yapısı	İÖB bir veri ile ilişkili olarak çalışıyorsa belirli bir veri yapısı kullanmak durumundadır.

İÖB öğesi Şekil 4.3.10.1. de görülen sembol ile temsil edilmektedir.

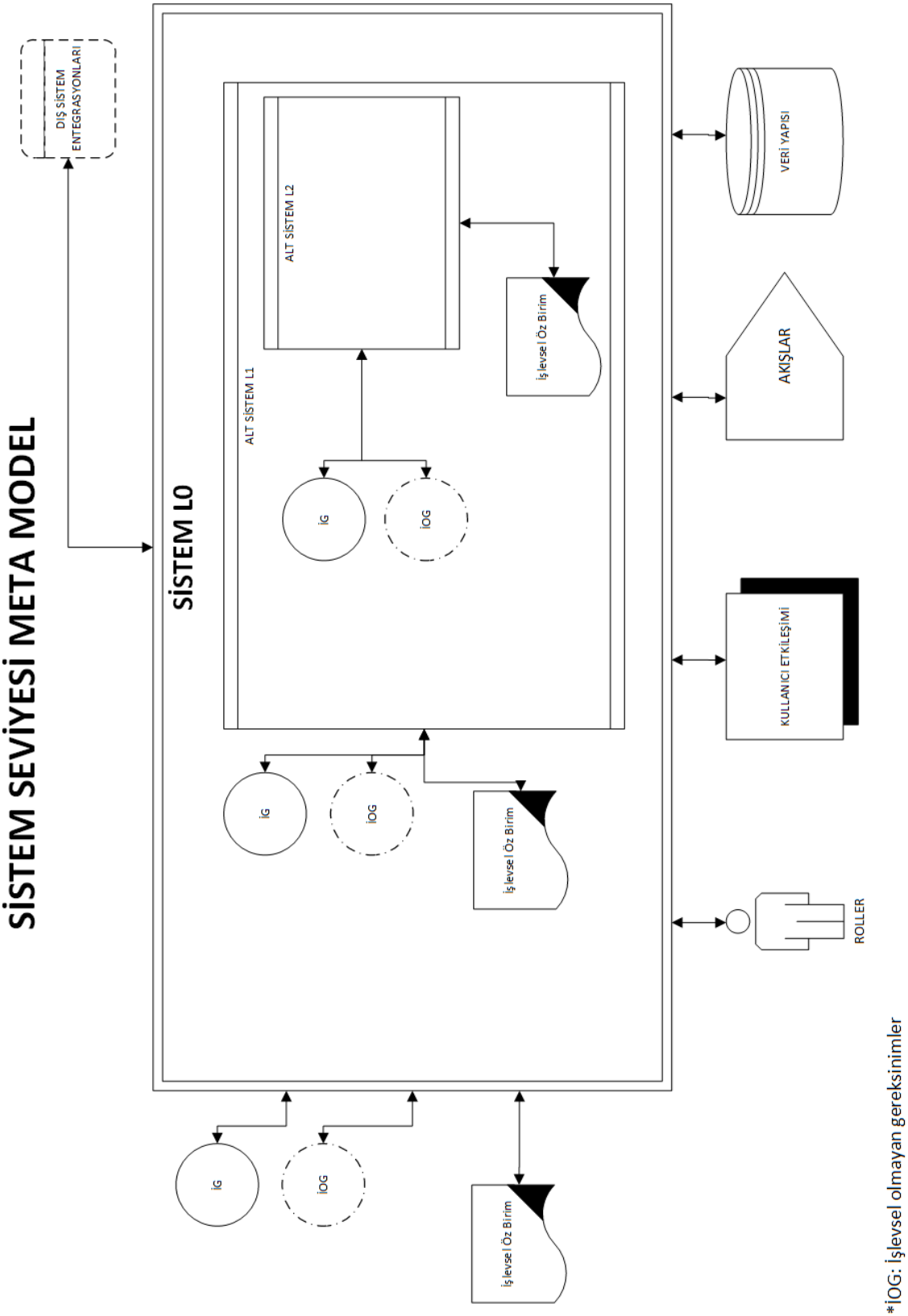


Şekil 4.3.10.1. İşlevsel Öz Birimin Sembolü

4.3.11. Modelin Genel Yapısı

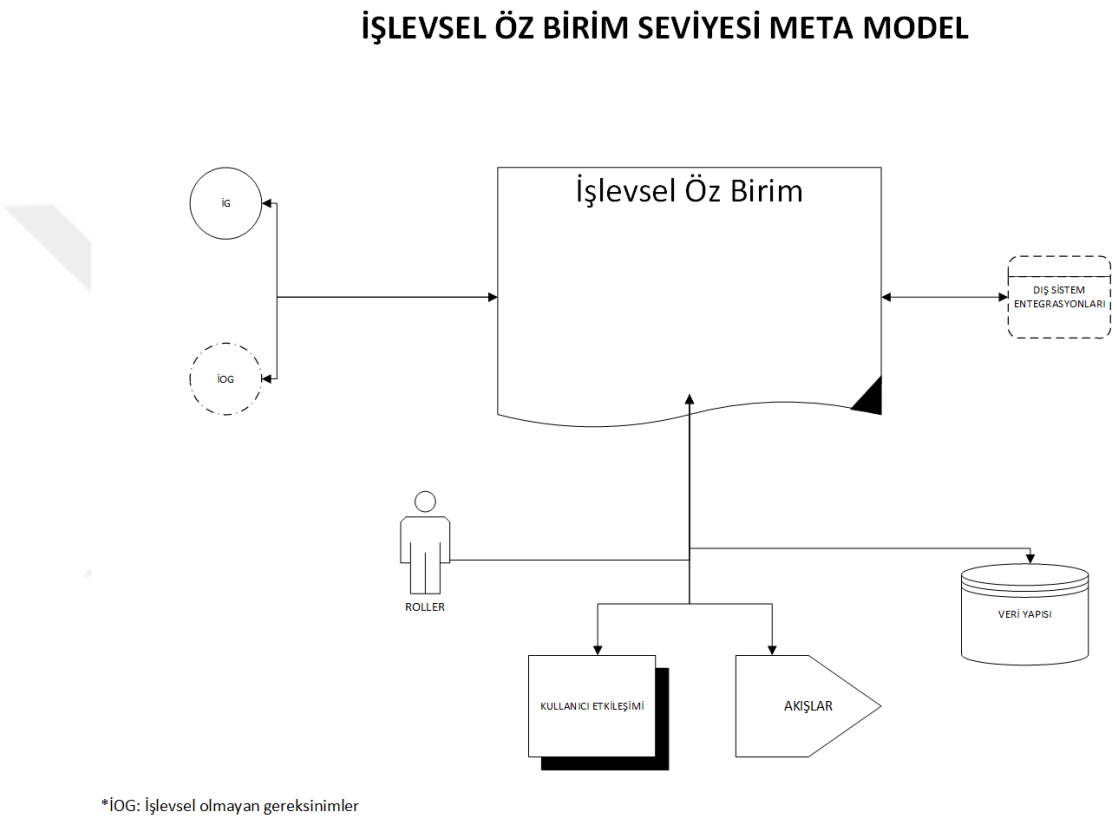
Model kademeli detaylandırmayı desteklemek üzere hiyerarşik seviyelendirmeye olanak tanıyan bir sistem – alt sistem yapısı çerçevesinde tasarlanmıştır. Bu seviyelendirmede gerektiği kadar alt sistem kullanılarak her türlü karmaşıklık seviyesindeki ve büyüklükteki yazılım için uygulanabilir bir model ortaya konmaya çalışılmıştır.

Yeni modelin meta-modeli iki seviyeli olarak yapılandırılmıştır. İlk seviye, sistem seviyesidir ve bu seviyede dış sistem entegrasyonu, sistem seviyesindeki işlevsel olmayan gereksinimler, amaçlar, roller, iş akışları, arayüzler, veri yapısı ve alt sistemler ile bunların birbirleriyle ilişkileri tanımlanmıştır.



Şekil 4.3.11.1 Sistem Seviyesi Meta Modeli

Sistem seviyesinde meta model Şekil 4.3.11.1. de gösterilmektedir. Meta-modelin ikinci seviyesi "işlevsel öz birim" (İÖB) yapısıdır. İÖB yapısında sistem ve alt sistem seviyelerinde tanımlanmış olan öğelerin, en atomik; tek başına anlamlı olabilecek en küçük analiz birimi, seviyede karşılıkları bir bütün öğe olarak ele alınmaktadır. İÖB ile ilişkili öğeler ise Şekil 4.3.11.2. de görülmektedir.



Şekil 4.3.11.2. İÖB Seviyesi Meta Modeli

4.4. Modelin Belirtimi

Çalışmayla ortaya koyulan yeni belirtim modelinin en önemli bileşeni "işlevsel öz birim" olarak tanımlanmıştır. İşlevsel öz birim sistemde gerçekleştirilen bir işlevin kullanıcı, oturum, işlem vb. açılardan anlamlı en küçük birimi olarak tarif edilmiştir. İşlevsel öz birimle birlikte işlevsel gereksinimlerin ölçeklenebilirliği sağlanmış, aynı zamanda iş akışı adımı olarak da işlevsel öz birimi kullanılarak analiz sonucu ortaya çıkan

akış bazlı gereksinimlerin standartlaştırılmasının sağlanması hedeflenmiştir. Aşağıdaki başlıklarda yeni modelin belirtiminde kullanılan dokümanlar ve öğeler açıklanmaktadır.

4.4.1. Sistem Perspektifi Dokümanı

Analiz süreci içerisinde ortaya çıkan ilk doküman "Sistem Perspektifi Dokümanıdır". Ön analiz aşamasının sonunda ortaya çıkan bu dokümanda yer alan başlıklar ve açıklamaları şu şekildedir:

1. Sistemin Adı: Başlık altında tanımlanan sisteme verilen isim yazılır. Eğer sisteme henüz bir isim verilmediyse de takma bir isimle ya da proje adımlarından biriyle isimlendirmekte fayda vardır.
2. Amaçlar: Buraya maddeler halinde sistemden beklenen fayda, sistemin yerine getirmesi gereken şeyler yazılır. Mümkünse önem ve öncelik sırasına göre yazılmalıdır. Sistem ve alt sistem öğelerinin detay bilgisi olarak tanımlanırlar.
3. Roller: Sistemi kullanacak olan kullanıcı grupları tanımlanır. Eğer grup adı yeterince açık değilse kısa bir açıklama eklenmelidir.
4. İş Akışları: Sistemde var olacak temel iş akışları burada listelenir. Her bir iş akışının ana amacı, başlangıç ve bitiş noktaları açıklanmalıdır.
5. İşlevsel Olmayan Gereksinimler: Bu bölüme sistemle ilgili önemli ve kritik işlevsel olmayan gereksinimler yazılır. Özellikle güvenlik, performans, kullanıcı deneyimi gibi işlevsel olmayan gereksinimlerle ilgili genel beklentinin alınması faydalı olacaktır.
6. Alt Sistemler: Burada akışlar da dahil olmak üzere sistemden beklenen temel işlevsel özellikler isimleriyle listelenir. Eğer işlevin adı işlevi açıklamıyorsa kısa bir açıklama da eklenebilir.
7. Dış Sistemler ve Entegrasyon: Bu başlık altına varsa sistemin dış sistemlerle entegrasyonları yazılır.
8. Doküman Bilgileri: Dokümanı hazırlayan, hazırlanma tarihi, versiyon, onaylayan paydaşlar gibi bilgileri içermektedir.

Örnek bir sistem perspektif dokümanı Şekil 4.4.1.1 de görülmektedir.

<table>
<tr><th colspan="5">Personel İzin Yönetimi – Sistem Perspektifi</th></tr>
<tr><td>Açıklamalar:</td><td colspan="2">• Personel izin sürecinin hatasız, minimum eforla en hızlı şekilde iletmek.”</td><td>Roller:</td><td>• Personel (tüm çalışanlar).
• Birim Yöneticisi (personelin iznini onaylamaya yetkili amiri).
• İK Uzmanı
• Sistem
• Sistem Yöneticisi</td></tr>
<tr><th>Akışlar</th><th colspan="2">İşlevsel Olmayan Gereksinimler</th><th>Alt Sistemler</th><th>Dış Sistemler ve Entegrasyon</th></tr>
<tr><td>• İzin talep yönetimi iş akışı</td><td colspan="2">• Sistem web uygulaması olarak çalışacaktır. Ve kurum dışından da erişilebilir olacaktır.
• Tüm sayfalar SSL üzerinde çalışmalıdır.</td><td>• İzin Talep Yönetimi
• İzin Bakiye Yönetimi
• İzin Raporları</td><td>• Active Directory, kullanıcı bilgileri ve bağlı oldukları yönetici bilgileri</td></tr>
<tr><td>NOTLAR:</td><td colspan="4"></td></tr>
<tr><td>Hazırlayan:</td><td colspan="2"></td><td colspan="2">Onaylayan Paydaşlar</td></tr>
<tr><td>Tarih:</td><td colspan="2"></td><td>Adı / Unvanı / Görevi</td><td>Tarih</td></tr>
<tr><td>Versiyon:</td><td colspan="2"></td><td></td><td></td></tr>
</table>

Şekil 4.4.1.1. Sistem Perspektif Dokümanı

4.4.3. Detay Analiz Dokümanı

Sistem ve alt sistemlerin detaylı olarak analiz edildiği aşama sonrasında ortaya çıkan doküman detay analiz dokümanıdır. Detay analiz dokümanı sistem ve alt sistemler hiyerarşisinde her seviyede tüm öğelerin detaylı bir şekilde gösterimi olarak ortaya konacaktır. Detay analiz dokümanının ana başlık yapısı şu şekildedir:

1. Proje Seviyesi,
    1.1. Proje Adı,
    1.2. Proje Detayları,
    1.3. Kullanıcı Rolleri,
    1.4. Dış Sistem Entegrasyonları,
    1.5. İşlevsel Olmayan Gereksinimler,
    1.6. İşlevsel Gereksinimler,
    1.7. Kullanıcı Etkileşimleri,
    1.8. Akışlar,
    1.9. Veri Yapıları,
    1.10. İşlevsel Öz Birimler,

1.10.1. İlişkili Roller,

1.10.2. İlişkili Entegrasyonlar,

1.10.3. İlişkili İşlevsel Gereksinimler,

1.10.4. İlişkili İşlevsel Olmayan Gereksinimler,

1.10.5. İlişkili Kullanıcı Etkileşimleri,

1.10.6. İlişkili Akışlar,

1.10.7. İlişkili Veri Yapıları,

1.11. Alt Sistemler,

1.11.1. Alt Sistem 1,

1.11.1.1. Alt Sistem Adı,

1.11.1.2. Alt Sistem Detayı,

1.11.1.3. İşlevsel Olmayan Gereksinimler,

1.11.1.4. İşlevsel Gereksinimler,

1.11.1.5. Kullanıcı Etkileşimleri,

1.11.1.6. Veri Yapıları,

1.11.1.7. Akışlar,

1.11.1.8. İşlevsel Öz Birimler,

1.11.1.8.1. İlişkili Roller,

1.11.1.8.2. İlişkili Entegrasyonlar,

1.11.1.8.3. İlişkili İşlevsel Gereksinimler,

1.11.1.8.4. İlişkili İşlevsel Olmayan Gereksinimler,

1.11.1.8.5. İlişkili Kullanıcı Etkileşimleri,

1.11.1.8.6. İlişkili Akışlar,

1.11.1.8.7. İlişkili Veri Yapıları,

1.11.2. Alt Sistem (N)…

4.5. Modelin İşletim Süreci

Ortaya konulacak belirtim modelini kullanan bir gereksinim analizi sürecini tarif etmek üzere yapılan literatür taramalarından sonra belirlenen analiz aktiviteleri aşağıdaki gibi sıralanmıştır:

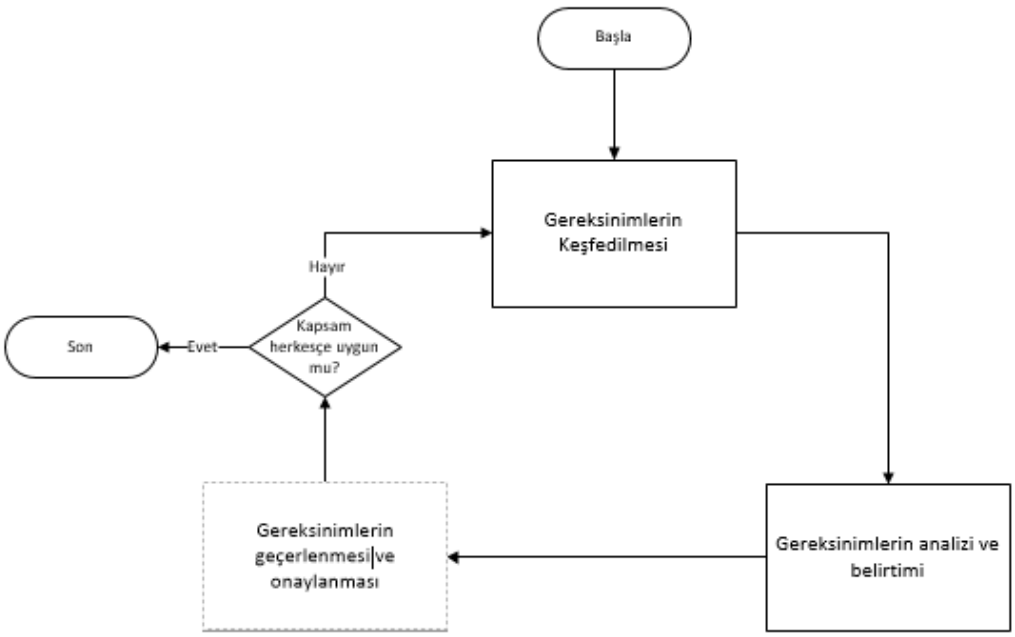
1. Gereksinimlerin meydana çıkartılması, toplanması, keşfedilmesi
2. Gereksinimlerin önceliklendirilmesi
3. Gereksinimlerinin birbirleriyle olan ilişkilerinin tarif edilmesi ve izlenmesi
4. Gereksinimlerle ilgili durumların (önceliklendirme, çatışma, kapsama dahil olma durumu) müzakeresi
5. Gereksinimlerin tüm paydaşlar tarafından anlaşılabilir olacak şekilde yeniden ifade edilmesi
6. Gereksinimlerin belirtimi, yazımı
7. Gereken durumlarda mevcut gereksinimlerden yeni gereksinimlerin türetilmesi
8. Gereksinimlerin sınıflandırılması (işlevsel, işlevsel olmayan, vb.)
9. Yüksek seviyeli gereksinimlerin detaylandırılarak çözümlenmesi
10. Gereksinimlerin geçerlenmesi
11. Gereksinimlerin doğrulanması
12. Kapsamın yönetimi
13. Gereksinimlerin onaylanması
14. Gereksinim değişikliklerinin yönetilmesi
15. Gereksinimlerin versiyon yönetimi
16. Gereksinimlerin ilgili paydaşlara dağıtımının yapılması ve iletişimi
17. Kavramsal tasarımın oluşturulması
18. Sistem büyüklük ve karmaşıklığının tahminlemesi

Bu listeye BABOK (Babok v3, 2015) ve daha birçok kaynakta bahsedilen paydaş yönetimi, kapsamın sınırlandırılabilmesi amacıyla alınmamıştır. Bunula birlikte analiz süreci tasarlanırken çevik yaklaşımların öne sürdüğü kademeli detaylandırma ve yinelemeli geliştirme esas alınarak analiz süreci de aşamalı ve yinelemeli olarak tasarlanmıştır.

Şöyle ki; analiz süreci "ön analiz" ya da "talep olgunlaştırma" fazıyla başlamakta ve bu aşamada sistemin amaçları, dış sistem entegrasyonları, sistemi kullanacak kullanıcı rolleri, temel iş akışları, önemli işlevsel olmayan gereksinimler ve öne çıkan alt sistemler tanımlanmaktadır. Ön analiz aşamasında sistemin kuş bakışı büyük resmi, yüksekten ve ana hatlarıyla çizilmektedir. Ön analiz aşaması kendi içinde de yinelemeli bir süreçle işletilmektedir.

Ön analiz aşaması belirli bir olgunluğa ulaşıp tüm paydaşlarca yeterli bulunduğunda "detay analiz" aşamasına geçilmektedir. Detay analiz aşamasında alt sistem detayına kadar inilerek tüm sistem öğeleri ve diğer sistem bileşenlerini ayrıntılı olarak tanımlayacak ve bileşenler arasındaki ilişkiyi ortaya koyacak şekilde çalışmalar yürütülür. Bu aşama da kendi içinde alt sistem bazında yinelemeli olarak gerçekleştirilmektedir.

Tüm bu yapının altında yatan temel kavramsal süreç Şekil 4.5.1. de görüleceği üzere gereksinimlerin keşfedilmesi, gereksinimlerin analizi ve belirtimi ile gereksinimlerin geçerlenmesi ve onaylanması başlıklarıyla tanımlanan yinelemeli şablondur.



Şekil 4.5.1. Gereksinim Analiz Süreci

Gereksinimlerin keşfi için bire-bir görüşme tekniğinden, çalıştay toplantılarına, odak gruplarından ankete kadar literatürde geçen her türlü yöntem kullanılabilmektedir. Yeni model bu görev için özel bir teknik tanımlamamaktadır. Gereksinimlerin keşfi aşamasında gerçekleştirilen alt görevler şu şekildedir:

1. Gereksinimlerin meydana çıkartılması,
2. Gereksinimlerle ilgili durumların müzakeresi,
3. Gereksinimlerin sınıflandırılması.

Gereksinimlerin analizi ve belirtimi görevi ise kendi içerisinde aşağıdaki alt görevleri içermektedir:

1. Gereksinimlerin önceliklendirilmesi,
2. Gereksinimlerinin birbirleriyle olan ilişkilerinin tarif edilmesi ve izlenmesi,
3. Gereksinimlerin tüm paydaşlar tarafından anlaşılabilir olacak şekilde yeniden ifade edilmesi,
4. Gereksinimlerin belirtimi ve yazımı,

5. Gereken durumlarda mevcut gereksinimlerden yeni gereksinimlerin türetilmesi,
6. Gereksinimlerin sınıflandırılması (işlevsel, işlevsel olmayan, vb.),
7. Gereksinimlerin doğrulanması.

Analist bu aşamada analiz dokümanını oluştururken ortaya konulan belirtim modelini kullanmalıdır. Diğer görevler içinse literatürde tanımlı herhangi bir modeli kullanabilir.

Son olarak, gereksinimin belirtimi de gerçekleştirildikten sonra, çalışmalar sonucu ortaya konan çıktılar ilgili paydaşlarca geçerli olup olmadığı sınanır ve geçerli olması durumunda yetkili bir paydaş tarafından gereksinimler için onay alınır.

Bu yinelemenin paralelinde çalıştırılacak analiz görevleriyse şöyledir:

1. Yüksek seviyeli gereksinimlerin detaylandırılarak çözümlenmesi,
2. Gereksinimlerin geçerlenmesi,
3. Kapsamın yönetimi,
4. Gereksinimlerin onaylanması,
5. Gereksinim değişikliklerinin yönetilmesi,
6. Gereksinimlerin versiyon yönetimi,
7. Gereksinimlerin ilgili paydaşlara dağıtımının yapılması ve iletişimi,
8. Kavramsal tasarımın oluşturulması,
9. Sistem büyüklük ve karmaşıklığının tahminlemesi.

Aşağıdaki başlıklarda gereksinim analiz süreçleri, süreç adımları ve çıktıları detaylı olarak anlatılmaktadır.

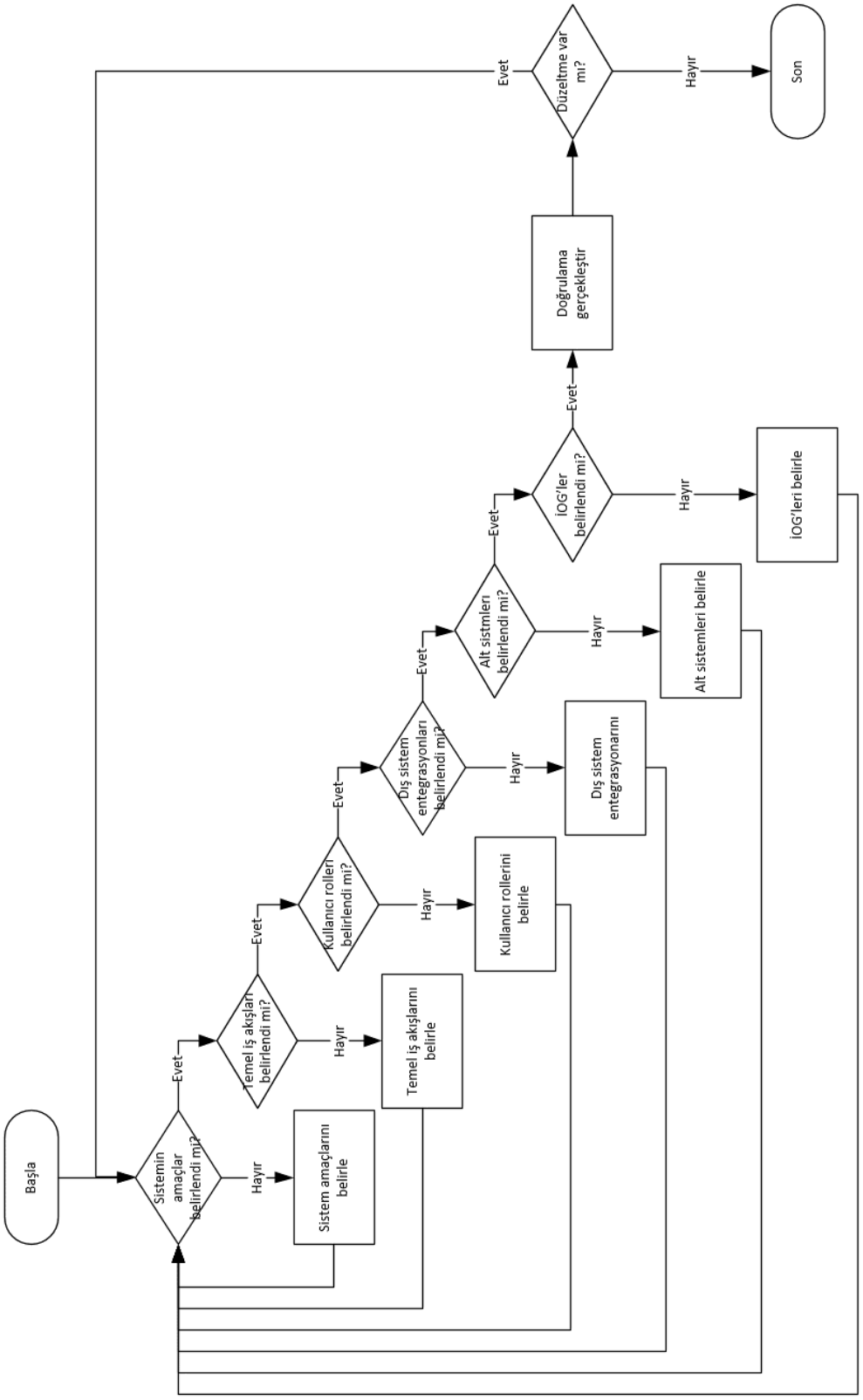
4.5.1. Ön Analiz Aşaması

Ön analiz aşaması, kilit paydaşlarla yapılacak gereksinim meydana çıkartma çalışmalarıyla ihtiyacın tespit edildiği, sistemin ana hatlarıyla sahip olması gereken işlevsel ve işlevsel olmayan gereksinimlerinin belirlendiği ve varsa temel iş akışlarının ortaya konduğu aşamadır. Bu aşamada önerilen model için analist aşağıdaki görevleri yerine getirmelidir:

1. Anahtar paydaşların sistemle ilgili beklentilerinin, sistemin amaçlarının belirlenmesi
2. Sistemdeki temel kullanıcı rollerinin tespiti
3. Sistemdeki temel iş akışlarının belirlenmesi
4. Varsa sistemin dış sistemlerle entegrasyonlarının tespiti
5. Sistemin ve alt sistemlerinin belirlenmesi
6. Sistemin önemli işlevsel olmayan gereksinimlerinin tespiti

Analist projenin en başında belirlenecek olan anahtar paydaşlarla görüşerek yukarıdaki bilgilere ulaşmaya çalışır ve bu bilgileri analiz edip, modelleyip dokümantasyonu yapılarak paydaşların onayına ve teyidine sunar. Bu adımlar ve süreç tüm kilit paydaşlarla görüşmeler tamamlanıp tüm kilit paydaşlar ve özellikle de karar vericiler ortaya konan kapsam ve temel gereksinim bileşenleri üzerinde hemfikir olana kadar devem etmektedir.

Süreç Şekil 4.5.1.1. de gösterildiği üzere sistemin amaçlarının tespitiyle başlamakta, temel iş akışları, kullanıcı rolleri, dış sistem entegrasyonları, alt sistem yapısı ve işlevsel olmayan gereksinimlerin meydana çıkartılması ve dokümantasyonuyla devam etmektedir.



Şekil 4.5.1.1. Ön Analiz Süreci

Sürecin sonunda, belirtilen öğelerle ilgili bir yenilik olup olmadığı kontrol edilir. Eğer yeni bir bilgi ya da değişiklik varsa süreç yinelenir. Eğer yeni bir bilgi veya düzeltme yoksa ortaya konulan bilgiler ilgili paydaşların onayına ve geçerlemesine sunulur.

Ön analizin tamamlanmasıyla ortaya "sistem perspektifi dokümanı" çıkmaktadır.

4.5.2. Detaylı Analiz Aşaması

Detaylı analiz aşaması, kilit paydaşlar dahil projeyle ilgili tüm paydaşların yer aldığı bir çalışmadır. Bu çalışmada "sistem perspektifi dokümanı" baz olarak alınır. Bu aşamada ön analiz dokümanıyla başlanan sistem öğelerinin tespiti genişletilerek tüm sistem öğeleri detaylıca ortaya konmaktadır. Detaylı analiz çalışmasında da ön analizde olduğu gibi yinelemeli bir şekilde gereksinimlerin keşfi, analizi, belirtimi, doğrulanması ve onaylanması ana adımları uygulamaktadır.

Önerilen modelde detay analiz aşamasında analist aşağıdaki görevleri yerine getirmelidir:

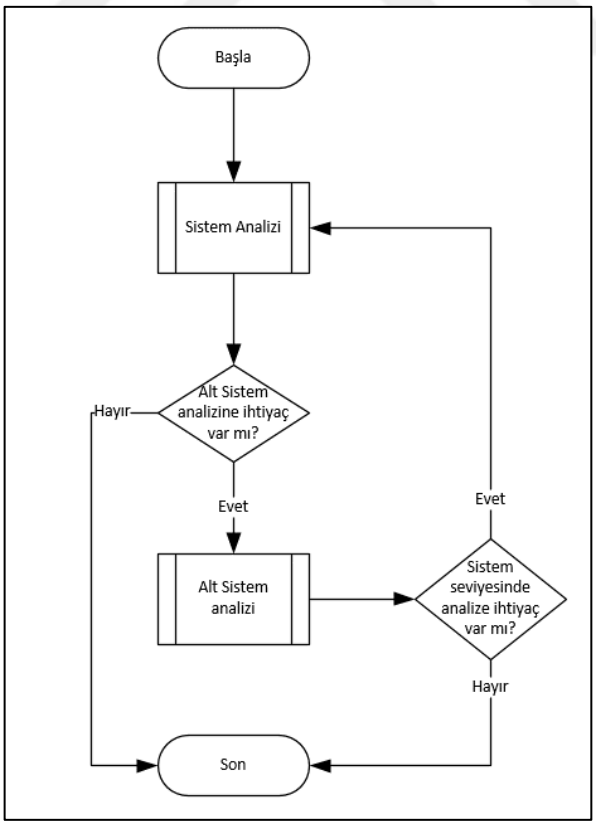
1. Sistemdeki tüm kullanıcı rollerinin tespiti ve rollerle ilgili detaylı bilginin ortaya konulması
2. Yeni paydaşlarla yapılan görüşmelerde ortaya çıkan yeni sistem amaçları varsa bunların ortaya çıkartılması
3. Dış sistemlerle ilişkili entegrasyonların detaylandırılması
4. Sistem seviyesindeki tüm akışların tespiti ve detaylandırılması
5. Sistemin tüm alt sistemlerinin belirlenmesi
6. Alt sistem seviyesindeki birincil ve ikincil senaryoların tespiti ve detaylandırılması
7. Sistemin tüm işlevsel olmayan gereksinimlerinin tespiti ve detaylandırılması
8. Sistemdeki alt sistemlerin tüm işlevsel öz birimlerinin tespiti ve detaylandırılması
9. Sistemdeki arayüzlerin tespiti ve detaylandırılması

10. Sistemin veri öğelerinin ve birbirleriyle ilişkilerinin tespiti

11. Veri öğelerinin detaylarının tespiti ve veri yapısının ortaya konması

Bu aşamada projenin tüm paydaşlarıyla gereksinim meydana çıkartma teknikleri kullanılarak gereksinim edinimi yapılır. Detay analiz yapılırken projenin türüne, paydaşların durumlarına göre analiz öğelerinin öncelikleri değişebilmektedir. Bununla birlikte temel olarak öncelikle iş akışlarına dayalı olarak alt sistemler detaylandırılmalı ve ardından sistem seviyesindeki arayüz, veri modeli, veri detayı gibi diğer öğelerin detaylandırılmasına geçilmelidir. Sistem seviyesinde iş akışı bulunmadığı durumlarda alt sistem detaylarına inilerek buradaki akışlar ele alınmaktadır. Böylece bir taraftan sistemle ilgili işlevsel özellikler meydana çıkartılırken, bir taraftan da bu işlevsel özelliklerin akışlarla ilgisi tarif ediliyor olacaktır.

Detaylı analiz süreci Şekil 4.5.2.1. de görüldüğü üzere sistem seviyesinde analiz ve alt sistem seviyesinde analiz olmak üzere iki yineleme yapısı içerisinde yürütülmektedir. Ayrıca bu yinelemeler arasında da alt yinelemeler gerçekleştirilerek kademeli derinleşmeye dayalı bir süreç izlenmektedir.



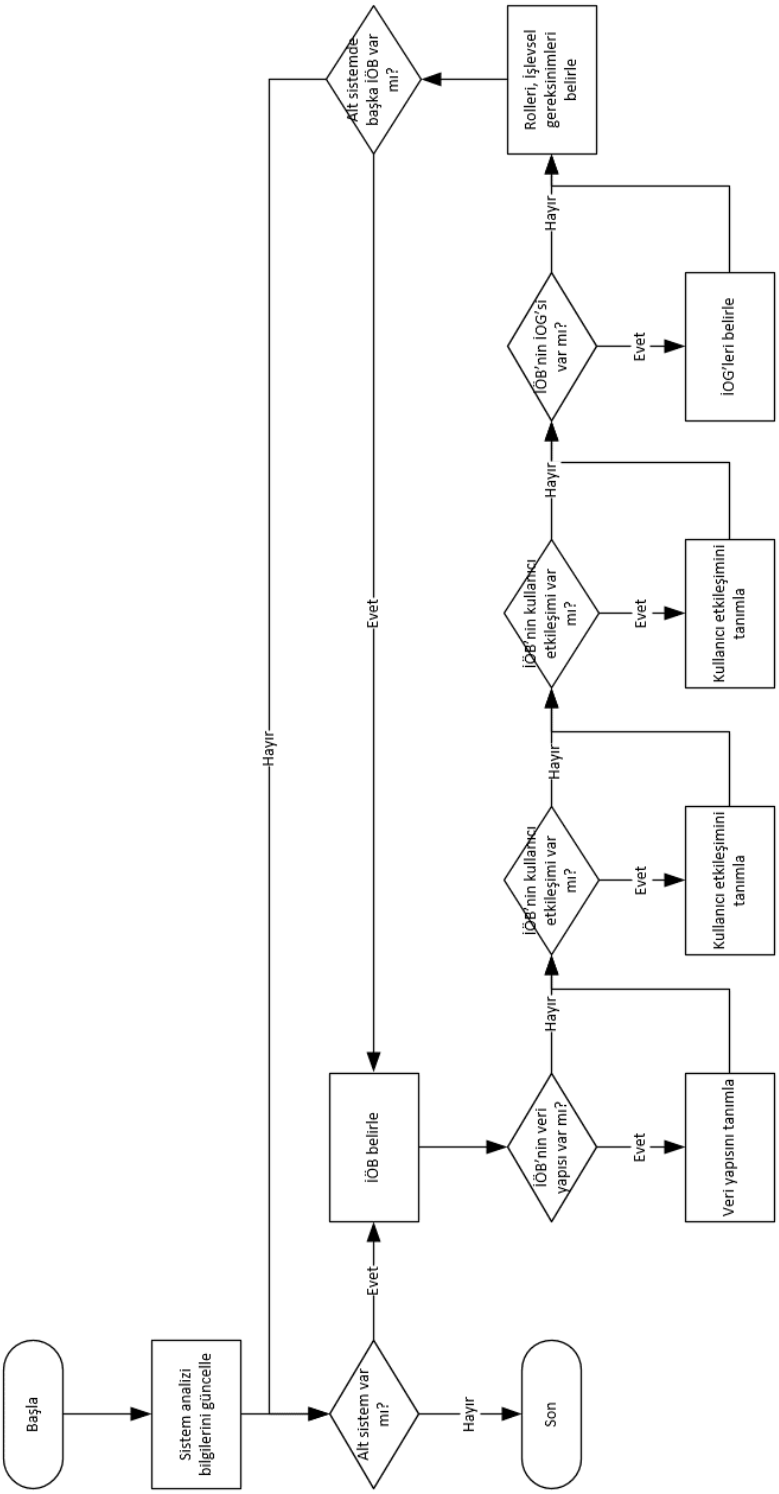
Şekil 4.5.2.1. Detay Analiz Süreci

Analist sistem seviyesinde analiz yaparken sistem perspektifi dokümanını referans alarak başlamaktadır. Analist öncelikle elindeki amaçlar, roller, iş akışları, işlevsel olmayan gereksinimler ve alt sistemlerle ilgili bilgiyle başlar ve bu bilgilerle ilgili sorular sorarak detaylandırır. Bu aşama yine analiz sürecinin öz kavramsal modelinin keşfetme kısmına denk gelmektedir.

Detay analiz sonrasında ortaya "gereksinim belirtim dokümanı" çıkmaktadır. Bu dokümanda ön analiz dokümanı başlıklarıyla birlikte iş akışları, veri modeli, veri yapısı, işlevsel olmayan gereksinimler, işlevsel öz birimler, senaryolar, entegrasyon, roller, amaçlar ve sistem-alt sistem başlıkları yer almaktadır.

Detay analizin sistem analizi aşamasında detaylandırılacak öğeler arasında katı ve net bir şekilde sıralama yapmak mümkün olmamakla birlikte öncelikle varsa amaçlarla ilgili güncellemeler ele alınmalı, amaçlar konusunda mutabık kalındıktan sonra işlevsel gereksinimlerin detaylandırılmasına geçilmelidir. İşlevsel gereksinimlere bağlı işlevsel öz birimlerin tespit edilebilmesi için önemlidir. İş akışları, iş kuralları, işlevsel gereksinimler ve diğer analiz bilgileri net bir şekilde detaylandırıldıktan sonra, sistemin yerine getireceği her bir atomik işlev için işlevsel öz birim tanımlanır.

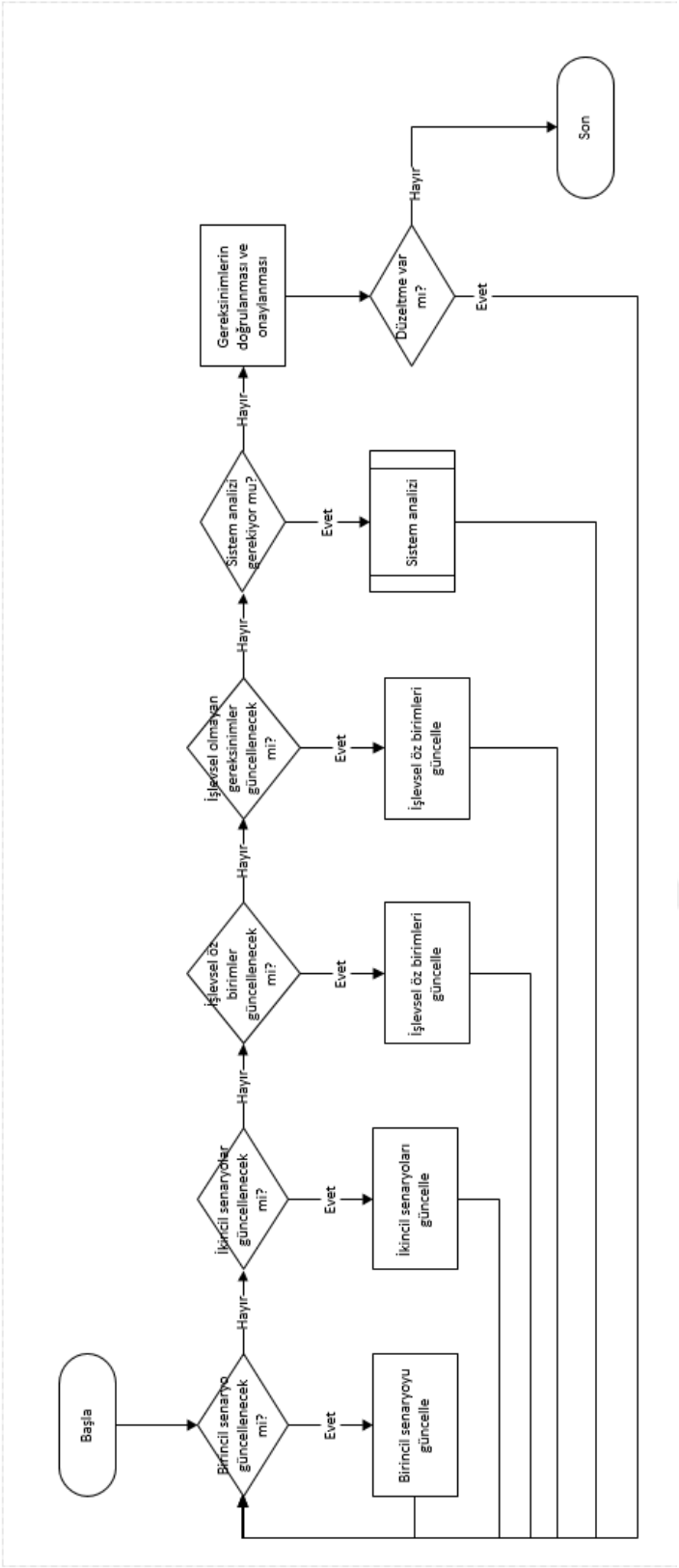
Sistem detay analiz süreci Şekil 4.5.2.2. de gösterilmektedir.



Şekil 4.5.2.2. Sistem Detay Analizi Süreci

İÖB'ler belirlendikten sonra sırasıyla arayüzler, veri modeli ve veri yapısı ele alınmalıdır. İlgili öğenin detaylandırılması sonrasında, daha önce ele alınan öğelerle ilgili güncellemeler söz konusu olabileceği için akış en başa yönlendirilmiştir. Analizin ilgili oturumunda tüm sistem seviyesindeki öğelerin güncellenmesi tamamlandığında alt sistem analizine geçilmektedir.

Alt sistem analizi de Şekil 4.5.2.3. de görüleceği üzere sistem analizi gibi yinelemeli bir süreç olarak tanımlanmaktadır ve bu aşamada modülün temel bileşenleri olan birincil ve ikincil senaryo ile işlevsel öz birimler detaylandırılmaktadır. Bu aşamadaki en önemli adım işlevsel öz birimin detay analizidir.



Şekil 4.5.2.3. İşlevsel Öz Birim Detay Analizi Süreci

Alt sistemin detay analizi tamamlandığında yapılan çalışmalar sonucunda sistem seviyesindeki öğelerden değişiklik ihtiyacı olup olmadığına bakılır ve değişiklik ihtiyacı varsa yeniden sistem analizi aşaması çalıştırılır.

4.6. Modelin Tahminleme Yaklaşımı

Ön-uç iş uygulamaları için yeni bir belirtim modeli oluşturulurken bu modelin aynı zamanda analiz edilen sistemle ilgili büyüklük ve karmaşıklığın ölçümlenebiliyor olmasıyla ilgili de çalışmalar yapılmıştır. Bu başlıkta modelin ölçümleme yaklaşımı, dayanağı ve uygulama tavsiyeleri yer almaktadır.

4.6.1. Ölçümleme Yaklaşımının Temel Öğeleri

Ortaya konulan model için çevik yöntemler başta olmak üzere birçok yazılım geliştirme yaklaşımının kullandığı çok sayıda ölçüt ve modelleme incelenmiş, kullanıcı senaryosu puanlama (user story point), kullanım vakası puanlama (use case point) yaklaşımları ve özel olarak da Andrew Pham ve Phuong-Van Pham (2012)'ın yazdıkları "Scrum in Action: Agile Software Project Management and Development" kitabındaki "An Objective Criteria-Based Estimating Process" yaklaşımın baz alınmasına karar verilmiştir.

Yeni model çevik yöntemlerde olduğu gibi öğrenmeye ve deneyime dayalı bir puanlamayı temel olarak kullanmaktadır. Bununla beraber, bu puanlamada öğrenmenin hızlı olması ve tutarlılığın artması için parametrik bazı öğelerin de kullanılması tercih edilmektedir.

Modelin merkezindeki en temel atomik öğe işlevsel öz birimdir. İÖB ile ilişkili tüm işlevsel ve işlevsel olmayan gereksinimler, işlevsel öz birim üzerinden puanlanır. Bu puan çevik yöntemlerdeki kullanıcı hikayesi puanı gibi göreceli bir puandır. Proje içindeki her bir öğenin diğer öğeye göre zorluk ve karmaşıklık değerini vermektedir.

Her bir işlevsel öz birimin emek tahminlemesinin daha doğru yapılabilmesi için, temel bazı parametreler üzerinden bulunan puanlar, parametrenin projeye özel ağırlık

katsayısıyla hesaplanır. Diğer taraftan karmaşıklık üzerinde işlevsel gereksinimler kadar etkili olabilecek işlevsel olmayan gereksinimler de göz önünde bulundurulmuş ve çarpan etkisiyle ölçümlemeye dahil edilmiştir.

Karmaşıklık faktörleri için değerler belirlenirken, çevik yöntemlerde genel olarak uygulanan Fibonacci sayısı kullanımı geleneği devam ettirilmiş ve 1, 2, 3, 5 ve 7 değerleri kullanılmıştır. Buradaki değerler adam-gün ya da adam-saat değildir. Bu değerler göreceli olarak büyüklük bildirmektedir. Örneğin puanı 3 olan bir işin kurumdaki ortalama emek 6 adam-saat ise puanı 5 olan bir işin kurumda tamamlanma süresi yaklaşık olarak 10 adam-saat olmalıdır.

4.6.2. Puanlama Faktörleri

Andrew Pham ve Phuong-Van Pham (2012) kitaplarında kullanıcı etkileşim türü, iş kurallarının karmaşıklığı, veri modeli yapısı, veri işleme operasyonu türü birer büyüklük ve karmaşıklık faktörü olarak ele alınmakta ve bunların her biri için ayrı ayrı 1 ile 3 aralığında değerler verilmektedir. Her bir kullanıcı hikayesi için bu değerler toplanarak bir "düzeltilmemiş puan" (unadjusted point) elde edilmekte ve daha sonra bu puana takım organizasyonu, geliştirme ortamı, teknoloji, süreç ve iş karmaşıklığı gibi boyutlar belirli bir oranda puanlanarak etki etmektedir.

Önerilen modelle ortaya konulacak olan gereksinim analizi sonucu tasarlanacak yazılımın karmaşıklık ve büyüklüğünün ölçümlenmesi için, modelin en kritik birimi olan işlevsel öz birim temel ölçümleme birimi olarak kabul edilmiştir. Bunun dışında kalan model öğelerinden sistem, alt sistem, dış sistem, entegrasyon, amaçlar, roller, iş akışları, birincil senaryo ve ikincil senaryolar ürün karmaşıklığı üzerinde direkt bir etkiye sahip olmadıkları düşünülerek hesaplama dışında bırakılmıştır. Çünkü bu öğelerin işlevsel karşılıkları işlevsel öz birim içerisinde gerçekleştirilmektedir.

Diğer sistem öğeleri olan işlevsel olmayan gereksinim, kullanıcı etkileşimi, veri modeli ve veri yapısı ise işlevsel öz birimin karmaşıklığını etkileyen faktörler olarak değerlendirilmektedir.

Önerilen modeldeki ölçümleme tekniği en küçük birim olarak işlevsel öz birimi esas alır ve ilk olarak işlevsel öz birim bazında aşağıdaki faktörlere bağlı olarak karmaşıklık ve büyüklük puanı hesaplanır:

1. İşlevsel öz birimin kullandığı arayüz, mesaj ya da bildirim (kullanıcı etkileşimi) karmaşıklığı,
2. İşlevsel öz birimin yaptığı işlem esnasında kullanacağı iş kuralı ve işlevsel gereksinimlerin karmaşıklığı,
3. İşlevsel öz birimin kullandığı veri modelinin, yapısının ve operasyonunun karmaşıklığı (ilişkili varlık sayısı ve kullanılan varlık alanları ve veri tabanında gerçekleştirilen işlemin karmaşıklığı).

İşlevsel öz birimin karmaşıklığı için yukarıda tanımlanan tüm bu faktörler için verilebilecek değerler ve bu değerler için kullanılabilecek referans kriterler metodu uygulayacak olan kurumca belirlenmeli ve gerçekleştirilen projelerden edinilen kazanımlarla iyileştirilmelidir. Bununla beraber bir başlangıç ya da örnek olması adına örnek projede uygulanan kriterler aşağıda verilmiştir.

Modeldeki karmaşıklık ve büyüklük faktörlerinden biri olan kullanıcı etkileşimi karmaşıklığı kullanıcı arayüz tasarımı, veri girişi doğrulama, arayüzle ilgili kurallar, mesaj gösterimi ve bildirim yollama gibi öğeleri ve işlemleri içermektedir. Bu öğeler çerçevesinde uygulanabilecek bir cetvel örnek olarak Çizelge 4.6.2.1. de gösterilmektedir.

Çizelge 4.6.2.1. Kullanıcı Etkileşimi Faktör Karmaşıklık Baz Puanı Belirleme Tablosu

Kullanıcı Etkileşimi Karmaşıklığı ve Büyüklüğü		
Karmaşıklık Tipi	**Açıklama**	**Değer**
Yok	Kullanıcı etkileşimi bulunmuyor.	0
Çok Basit	Sadece birkaç (en fazla 5) ekran öğesinden oluşan ve/veya tasarım emeği çok düşük olan arayüz ya da en fazla 3 cümleden oluşan mesaj ya da bildirim.	1
Basit	Düşük tasarım emeği gerektiren ve/veya az sayıda ekran öğesi (6 – 10) içeren arayüz ya da bir paragrafla sınırlı ve karmaşık biçimlendirmesi olmayan bildirim ya da mesajı.	2
Ortalama	Çok fazla olmasa da tasarım emeği gerektiren ve/veya 11 ile 20 arasında ekran öğesi olan arayüz ya da kullanıcı giriş doğrulaması emek gerektiren etkileşimler.	3
Karmaşık	Hem tasarım emeği hem de ekran öğesi sayısı fazla olan ve/veya girdilerin doğrulanması için algoritma tasarlanması gereken etkileşimler.	5
Çok Karmaşık	Yeni teknoloji kullanılan, karmaşık kullanılabilirlik gereksinimi içeren etkileşimler.	10

Uygulamada sadece iş kurallarının çalıştırıldığı, arayüz ve bildirim olmayan bir işlevsel öz birim mevcutsa, bu durumda arayüz karmaşıklık puanı olarak “0” verilmelidir.

İşlevsel gereksinimlerde iş kuralları, hesaplama, operasyon gibi unsurların karmaşıklığı esas alınmıştır. İş kuralı ve birim karmaşıklığı için Çizelge 4.6.2.2. de ifade edildiği gibi 5 seviye belirlenmiştir.

Çizelge 4.6.2.2. İşlevsel Gereksinim Faktör Karmaşıklık Baz Puanı Belirleme Tablosu

İşlevsel Gereksinim Karmaşıklığı ve Büyüklüğü		
Karmaşıklık Tipi	**Açıklama**	**Değer**
Yok	Herhangi bir iş kuralı yok.	0
Çok basit	Çok basit hesaplamalar ve koşul yapıları	1
Basit	Basit hesaplamalar ve koşul yapıları	2
Ortalama	Basit olmayan ama çok karmaşık da olmayan hesaplamalar ve koşul yapıları	3
Karmaşık	Çok sayıda iş kuralı var ya da iş kuralı çok karmaşık.	5

Veri operasyonu karmaşıklık ve büyüklük faktörü, işlevsel öz birimin kullanacağı veri öğeleri (tablolar), bunlar arasındaki ilişkiler, kullanılacak alanların sayısı ve gerçekleştirilecek veri operasyonun sayısı ve karmaşıklığı olarak Çizelge 4.6.2.3. de görüldüğü üzere 6 seviyeli bir yapıda belirlenmiştir.

Çizelge 4.6.2.3. Veri Modeli ve Yapısı Karmaşıklık ve Büyüklük Puanları Tablosu

Veri Modeli ve Yapısı Karmaşıklığı ve Büyüklüğü		
Karmaşıklık Tipi	**Açıklama**	**Değer**
Yok	Veri tabanı öğesine bağlı bir işlev değil	0
Çok Basit	Sadece bir varlık ve/veya en fazla 5 alan, ilişkili varlık yok, temel veri tabanı operasyonu.	1
Basit	2 adet varlık ve/veya en fazla 10 alan, basit ilişkili varlık, temel veri tabanı operasyonu.	2
Ortalama	3 adet varlık ve/veya en fazla 15 alan ve/veya en fazla 3 varlık ilişkisine kadar. Veri tabanı operasyonu basit olmayan ya da birden fazla olan veri işlemleri.	3

Karmaşık	4 ve üzerinde varlık ve/veya 16 ve üzerinde alan ve/veya 4 ve üzerinde varlık ilişkisi ve/veya karmaşık veri operasyonları.	5
Çok Karmaşık	Çok sayıda öğenin çok karmaşık yapıya ve ilişkilere sahip olduğu ve/veya karmaşık algoritmalar gerektiren veri operasyonları.	10

Buraya kadar tanımlanan faktörler İÖB puanlanması için baz oluşturacak öğelerdir. Puanlama faktörleri her bir İÖB için ayrı ayrı hesaplanır. Bu hesaplama sırasında bir İÖB için aynı faktörden birden fazla öğe varsa bunların her biri toplanarak o faktör için toplam bir puan oluşturulur. Örneğin bir İÖB hem bir ekran üzerinden çalışıyor hem belirli durumlarda belirli mesajlar gösteriyor hem de duruma göre bildirim yolluyorsa, bu üç kullanıcı etkileşimi faktörü için ayrı ayrı puan verilir ve bunların toplamı ilgili İÖB için tek bir faktör ham puanı olarak kullanılır.

Bulunan faktör ham puanı çizelge 4.6.2.4. de verilen örneğe benzer şekilde projeye uygun olarak belirlenmiş ağırlık katsayıları ile çarpılarak her bir İÖB için baz puanı hesaplanır.

Çizelge 4.6.2.4. Faktör Ağırlık Katsayıları Tablosu

Faktör Ağırlık Katsayıları		
Karmaşıklık Tipi		**Katsayı**
Kullanıcı Etkileşimi		1
İşlevsel Gereksinimler		1,2
Veri Yapısı		0,8

4.6.3. İÖB Faktörleri İçin Faktör Karmaşıklık Puanı Hesaplama

Model temelde İÖB'lerin belirli faktörler üzerinden puanlanmasına dayanmaktadır. Her bir İÖB her bir faktörden ilişkili olduğu öğe sayısı kadar bir ya da daha fazla "Faktör Karmaşıklık Baz Puanı" almaktadır. Örneğin bir İÖB'nin 3 tane işlevsel gereksinimle ilişkisi varsa. Bu işlevsel gereksinimlerin "İşlevsel Gereksinim Faktör Karmaşıklık Baz Puanı Belirleme Tablosu"ndan baz puan karşılıkları bulunur ve toplanarak İÖB için "Faktör Karmaşıklık Ham Puanı" (FKHP) bulunmuş olur. Bu tüm faktörler için ayrı ayrı gerçekleştirilir.

İÖB'lerin her biri için her bir faktörün FKHP bulunduktan sonra bu puanlar ilgili faktörün "Faktör Ağırlık Katsayısı (FAK)" ile çarpılır. Böylece ilgili İÖB için belirli bir faktörün katsayı uygulanmış İOG uygulanmadan öneki "Faktör Karmaşıklık Puanı (FKP)" bulunmuş olunur.

4.6.4. İÖB Net Karmaşıklık Puanı Hesaplama

FKP'ları bulunan İÖB'lerin FKP leri toplanarak İÖB'lerin "İÖB Karmaşıklık Ham Puanı" (İÖBKHP) bulunur. Böylece her bir İÖB için İOG katsayısı uygulanmadan önceki puanlama yapılmış olur.

Bu noktada İOG ile ilişkilendirilmiş olan İÖB'ler ilişkili oldukları İOG'ler esas alınarak "İşlevsel Olmayan Gereksinim Katsayıları" tablosundan ilgili puanlarla çarpılır ve böylece" İÖB Net Karmaşıklık Puanı (İÖBNKP)" bulunmuş olur.

İOG için önerilen ağırlık puanlaması Çizelge 4.6.4.1. de gösterilmektedir.

Çizelge 4.6.4.1. İOG Katsayıları Tablosu

İşlevsel Olmayan Gereksinim Katsayıları		
Karmaşıklık Tipi	**Açıklama**	**Çarpan**
Yok	Herhangi bir İOG ile ilişkili değil	1
Çok basit	Çok basit kullanılabilirlik gereksinimleri	1,05
Basit	Basit kullanılabilirlik, performans ya da güvenlik gereksinimleri	1,10
Ortalama	Orta seviyede kullanılabilirlik, güvenlik ve/veya performans gereksinimleri	1,50
Karmaşık	İleri seviyede kullanılabilirlik gereksinimleri, karmaşık güvenlik ve yüksek performans gereksinimleri	2
Çok Karmaşık	İşlem güçlüğü yüksek, ekibin tecrübesinin olmadığı çok sayıda unsur içeren İOG	4

4.6.5. Proje Toplam Karmaşıklık Puanı Hesaplama

"Proje Toplam Karmaşıklık Puanı (PTKP)" hesaplamak için projedeki tüm İÖB'lerin İÖBNKP'ları toplanır ve buna varsa entegrasyon puanları eklenir.

Entegrasyonlar sistem seviyesinde tanımlanacağından, entegrasyonlar için tamamen bağımsız bir puanlama oluşturulması ve oluşturulan bu puanın İÖB puanlarının toplamına eklenmesiyle toplan proje tahminlemesinin tamamlanması ön görülmüştür. Buradaki puanlar İÖB net puanına eş değerdir. Entegrasyon için önerilen puanlama tablosu Çizelge 4.6.5.1. de verilmiştir.

Çizelge4.6.5.1. Dış Sistem Entegrasyonu Puanı Tablosu

Dış Sistem Entegrasyon Puanı Tablosu		
Karmaşıklık Tipi	**Açıklama**	**Değer**
Yok	Sistemin herhangi bir başka sistemle entegrasyonu yok	0
Çok Basit	Çok basit, bilinen, tecrübe edilmiş entegrasyonlar, daha çok veri okumaya yönelik entegrasyonlar.	5
Basit	Ekibin aşina olduğu basit seviyede entegrasyonlar.	10
Ortalama	Veri gönderimi, alımı yapılan, birden fazla fonksiyonla çalışılan entegrasyonlar.	20
Karmaşık	Çok sayıda fonksiyonla çalışılan, belirli seviyede hâkim olunan entegrasyonlar.	50
Çok Karmaşık	Teknik olarak deneyim sahibi olunmayan ya da çok sayıda fonksiyonla karmaşık işlemin yapıldığı entegrasyonlar.	100

4.6.6. Proje Ekibinin Proje Karmaşıklık Puanına Etkisi ve Toplam Emek Tahmini

Yazılım projeleriyle ilgili tahminleme yapılırken göz önünde bulundurulması gereken önemli bir konu da projenin karmaşıklık büyüklüğüyle ne kadar adam-günde yapılacağını gösteren emek büyüklüğünün farklı konular olmasıdır. Her ne kadar projenin toplam emeği projenin toplam karmaşıklığıyla doğrudan orantılıysa da projeyi deneyimli ya da deneyimsiz bir ekibin yapacak olmasına bağlı olarak değişeceğinden modelin kurgusunda bu hesaplamanın ayrı tutulmasına karar verilmiştir.

Modelin karmaşıklık ve emek büyüklüklerinin ayrı ayrı yönetilmesi, kurum içinde ve belki aynı modeli uygulayan farklı kurumlarda birbirleriyle karşılaştırılabilecek proje karmaşıklık büyüklükleri oluşturabilmektir. Böylece tahminleme konusunda kurumların kendini iyileştirmeleri daha fazla örnek üzerinden öğrenme sağlanabileceğinden daha kısa sürelerde mümkün olabilecektir.

Diğer taraftan uygulamanın emek tahminlemesinin uygulamayı gerçekleştirecek olan proje ekibinin performans durumu göz önünde bulundurularak, emek tahminlemesi de gerçeğe daha yakın yapılmış olacaktır. Bunu gerçekleştirebilmek için de projede kullanılacak teknolojiye, dile, platforma ve projenin iş alanına takımın yatkınlığına bakılarak bir katsayı verilmelidir. Bu katsayı için karmaşıklık puanını bire-bir yansıtacak olan rakamı baz olarak ve kurumun ortalama performans değeri olarak "1 (bir)" katsayısı kullanılmalıdır. Eğer proje ekibinin performansının kurumun performansının %10 altında olduğu düşünülüyorsa katsayı olarak "1,10", %20 daha altında olduğu düşünülüyorsa "1,20" katsayıları kullanılabilir. Benzer şekilde proje ekibinin performansı %10 daha yüksekse katsayı olarak "0,90" ya da %20 daha iyiyse "0,80" kullanılabilir.

Ancak bunun öncesinde her bir karmaşıklık puanının kaç adam-saat olduğunun belirlenmesi gerekmektedir. Bunun için kurumlar tamamlanmış yazılım projeleri üzerinde çalışma yaparak eski analizleri İÖB modeline dönüştürüp, projelerin gerçekleşmelerine bakarak 1 karmaşıklık puanının kaç adam-saate denk geldiğini belirlemelidir. Bizim yaptığımız çalışmalarda her "bir karmaşıklık puanının" "bir adam-saat" olduğu ön görülmüştür ve yapılan örnek projelerde bu tahminin gerçeğe çok yakın olduğu tespit edilmiştir.

Tüm bu hesaplamaların sonunda proje tahmini emeği şu formülle hesaplanabilecektir.

Toplam Emek = PTKP x KPAGDK x BEBK

Formüldeki PTKP, proje toplam karmaşıklık emek puanını, KPAGDK karmaşıklık puanı adam-gün ya da adam-saat dönüşüm katsayısını ve BEBK ise proje ekibi başarım katsayısını ifade etmektedir.

4.6.7. Proje Toplam Karmaşıklık Puanı Hesaplama Özeti

Güncellenen ve iyileştirilen karmaşıklık hesaplama yöntemi bir algoritma ile ifade edilmesi gerekirse aşağıdaki gibi ele alınabilmektedir:

1. İlk olarak sistemdeki tüm İÖB'lerin net karmaşıklık puanı hesaplanır (İÖBNKP)
    1.1. Her bir İÖB teker teker ele alınır,
        1.1.1. İÖB'nin her bir faktörü için karmaşıklık puanı hesaplanır (FKP)
            1.1.1.1. Faktörün baz puan tablosundan ilgili değer alınır
            1.1.1.2. Faktörün baz puanı ağırlık tablosundaki katsayı ile çarpılır ve sonuç faktör puanına eklenir
        1.1.2. Varsa İÖB için İOG katsayısı kullanılarak İÖBNKP hesaplanır
    1.2. Tüm İÖB lerin İÖBNKP'leri toplanır.
2. Sistemin varsa entegrasyon puanı hesaplanır (EP)
3. Tüm İÖBNKP'ler toplanıp üzerine EP eklenerek toplam projenin toplam karmaşıklığı bulunur (PTKP)
4. Projede çalışacak ekibin başarım katsayısı ile PTKP çarpılarak emek puanı (PTKEP) bulunmuş olur.

4.7. Modelin Kullanımını Desteklemek Üzere Geliştirilen Yazılım

Sunulan modelinin etkin kullanımını sağlamak amacıyla, geresinim analizi belirtim modelinine ek olarak süreç ve tahminleme yönlerini de kapsayan bir yazılımla desteklenmesinin doğru olacağına karar verilmiştir. Bu çerçevede çalışmaya dahil olacak yazılımın kapsamı belirlenmiş, kapsam çerçevesinde temel gereksinimleri netleştirilerek yazılım geliştirilmiş ve güncellenen yeni modeli destekleyecek bir yapıda ortaya konulmuştur. Aşağıdaki başlıklarda yazılımın gereksinimleri, alt yapısı, ana ekranları ve kullanımına yönelik bilgiler verilmektir.

4.7.1. Yazılımın Kapsamı, Temel Gereksinimleri ve Fonksiyonları

Yazılımın temel amacı bu çalışmayla ortaya koyulan yeni gereksinim belirtim modelinin otomasyon ile dokümantasyon, tahminleme ve doğrulama yönlerinden desteklenmesidir. Bu amaçla geliştirilen yazılım için aşağıdaki yüksek seviyeli gereksinimler esas alınmıştır:

1. Yazılım, modeldeki tüm öğelerin hiyerarşik bir yapıda oluşturulmasını, güncellenmesini, silinmesini ve listelenmesini sağlayabilmelidir,
2. Yazılımda modele uygun şekilde öğeler birbirleriyle ilişkilendirilebilmelidir,
3. Yazılım, girilen bilgileri esas alarak sistem perspektif dokümanı oluşturabilmelidir,
4. Yazılım, girilen bilgileri esas alarak detay analiz dokümanı yazdırabilmelidir,
5. Yazılımla güncellenen modelin tahminleme mekanizması otomatik olarak çalışabilmelidir,
6. Yazılım, girilen öğeleriyle ilgili temel doğrulama kurallarını kontrol edebilmeli ve varsa hataları listeleyerek analistin daha efektif çalışmasına olanak sağlamalıdır.

Belirlenen bu gereksinimlerden altıncı madde olan doğrulamaların iş kuralları şu şekilde belirlenmiştir:

1. Her bir sistem ya da alt sistemin hiyerarşik yapı içinde en az bir İÖB içermesi gerekmektedir,
2. Her bir İÖB en az bir işlevsel gereksinim ile ilişkilendirilmelidir,
3. Her bir İÖB en az bir rol ile ilişkilendirilmelidir,
4. Her bir rol en az bir İÖB ile ilişkili olmalıdır,
5. Veri yapısı İÖB ler uyarılar listesinde gösterilmelidir,
6. Herhangi bir kullanıcı etkileşimiyle ilişkilendirilmeyen İÖB'ler uyarılar listesinde yer almalıdır.

4.7.2. Yazılım Geliştirmek İçin Kullanılan Dil, Teknoloji ve Mimari

Bir önceki başlıkta belirlenen yazılımı geliştirmek için kullanım pratikliği, internet bağlantısı ya da bir sunucu olmasına gerek kalmaksızın bağımsız olarak çalışabilmesi nedeniyle Microsoft teknolojileri kullanılarak Windows Uygulaması geliştirilmesine karar verilmiştir.

Kodlama için Microsoft .NET Framework 4.6.1 alt yapısı, Visual Studio 2017 üzerinden C# programlama dili tercih edilmiştir. Veri tabanı olarak ilişkisel veri tabanı olan MS SQL Express Edition 2016 sürümü tercih edilmiştir. Ücretsiz olan bu sürümde bazı kısıtlamalar olmakla beraber 10 GB veri saklama, 4 çekirdek ve 32 GB bellek destekleyebilmesi uygulama için yeterli olacağı düşünülmüştür.

Ayrıca uygulamanın geliştirme aşamasında GitHub platformu kodların tutulması ve versiyon takibi için kullanılmış ve geliştirme sonunda herkesin kullanımına açılmıştır. Kodların son sürümüne "https://github.com/kadircam/CFUWin" bağlantısı üzerinden ulaşılabilecektir.

4.7.3. Yazılımın Veritabanı Yapısı

Yazılımın veritabanı 12 adet tablodan oluşmaktadır. Bu tabloların isimleri, alanları ve ne amaçla kullanıldıkları Çizelge 4.7.3.1. de açıklanmaktadır.

Çizelge 4.7.3.1. Yazılımın Veri Tabanları Tablosu

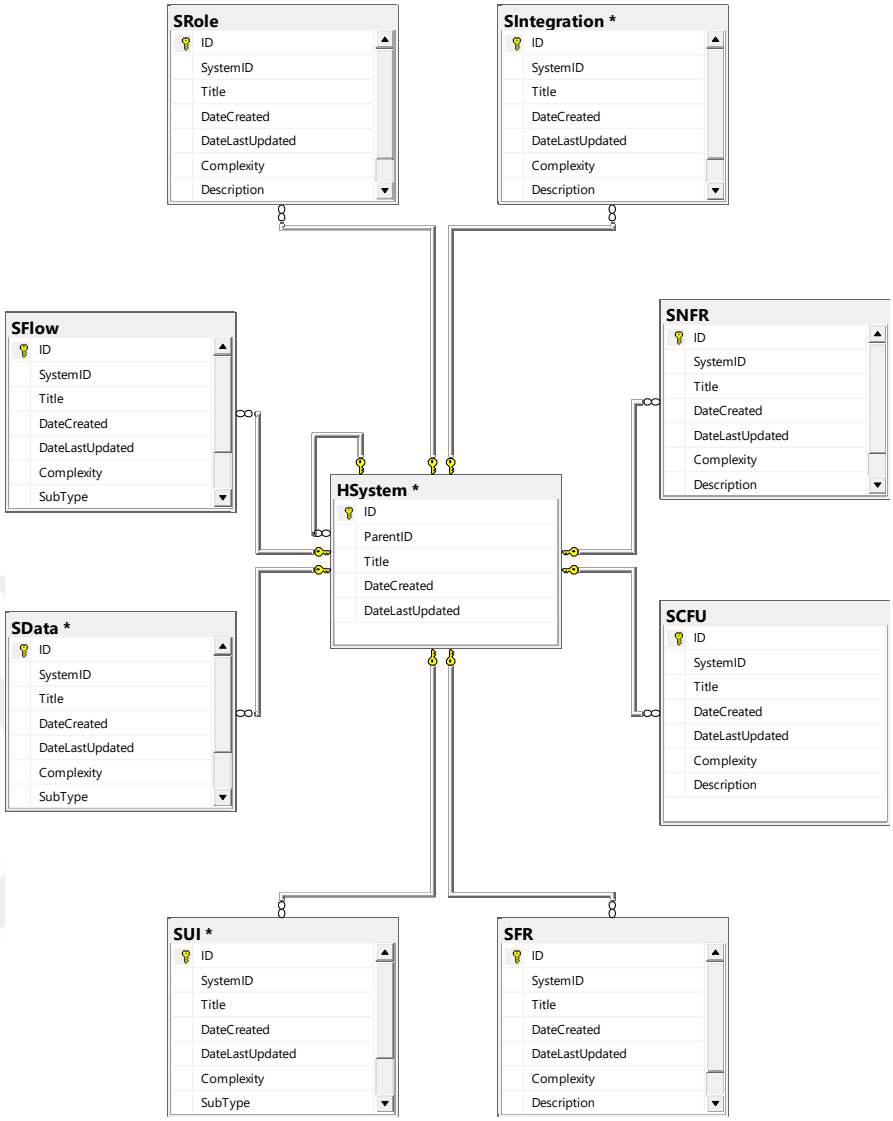
SN	Tablo Adı	Alanları	Açıklama
1	HAttachment	• [ID] • [ItemType] • [ObjectID] • [FileName] • [FileData]	Yazılımda yer alan modelin tüm öğelerine n adet dosya eklemek amacıyla kullanılmaktadır.
2	HSystem	• [ID]	En üst seviyede proje, alt seviyede alt sistemler olmak üzere,

		• [ParentID] • [Title] • [DateCreated] • [DateLastUpdated] • [Description]	birbirleriyle ilişkili bir şekilde sistemlerin tanımlanması amacıyla kullanılmaktadır. Kendine ParentID üzerinden referansı bulunmaktadır.
3	LCFUItem	• [ID] • [CFUID] • [ItemID] • [ItemType]	Sistem öğesi hariç diğer tüm öğelerin İÖB ile ilişkisinin tutulduğu tablodur.
4	SCFU	• [ID] • [SystemID] • [Title] • [DateCreated] • [DateLastUpdated] • [Complexity] • [Description]	İşlevsel öz birimlerinin tutulduğu tablodur.
5	SData	• [ID] • [SystemID] • [Title] • [DateCreated] • [DateLastUpdated] • [Complexity] • [SubType] • [Description]	Data yapılarının tutulduğu tablodur.
6	SFlow	• [ID] • [SystemID] • [Title] • [DateCreated] • [DateLastUpdated] • [Complexity] • [SubType]	İş akışı, birincil ve ikincil senaryoların tutulduğu tablodur.

		• [Description]	
7	SFR	• [ID] • [SystemID] • [Title] • [DateCreated] • [DateLastUpdated] • [Complexity] • [Description]	İşlevsel gereksinimlerin tutulduğu tablodur.
8	SIntegration	• [ID] • [SystemID] • [Title] • [DateCreated] • [DateLastUpdated] • [Complexity] • [Description]	Dış sistem entegrasyonlarının tutulduğu tablodur.
9	SNFR	• [ID] • [SystemID] • [Title] • [DateCreated] • [DateLastUpdated] • [Complexity] • [Description]	İşlevsel olmayan gereksinimlerin tutulduğu tablodur.
10	SRole	• [ID] • [SystemID] • [Title] • [DateCreated] • [DateLastUpdated] • [Complexity] • [Description]	Yapılan analizdeki rol öğelerinin tutulduğu tablodur.
11	SUI	• [ID]	Kullanıcı etkileşimi bilgilerinin tutulduğu tablodur.

		• [SystemID] • [Title] • [DateCreated] • [DateLastUpdated] • [Complexity] • [SubType] • [Description]	
12	HParameter	• [ID] • [ParamName] • [ParamValue]	Sistem ve tahminleme parametrelerinin tutulduğu tablodur.

Tablolardan HParameter, LCFUItem ve HAttachment hariç tamamı Şekil 4.7.3.1. de görüldüğü üzere HSystem ile ilişkilidir.

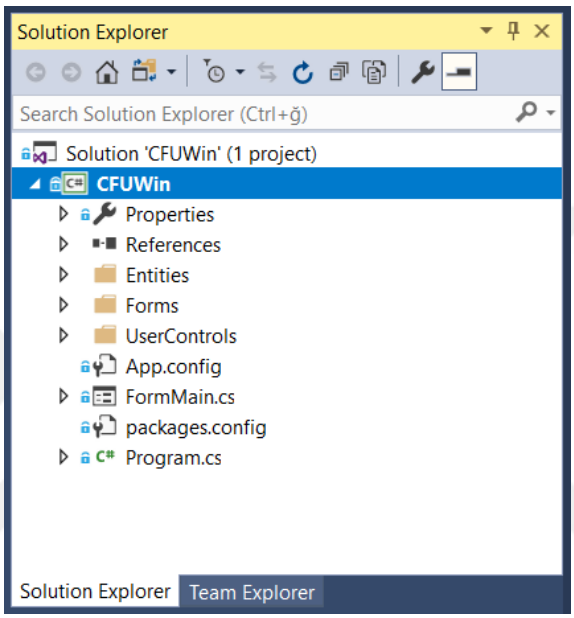


Şekil 4.7.3.1. Tablolar Arası İlişkiler Diyagramı

Buna ek olarak SData, SFlow, SFR, SIntegration, SNFR, SRole ve SUI tabloları LCFUItem tablosu aracılığıyla n-n ilişki kuracak şekilde SCFU ile ilişkilidir.

4.7.4. Kodlama Çalışmaları

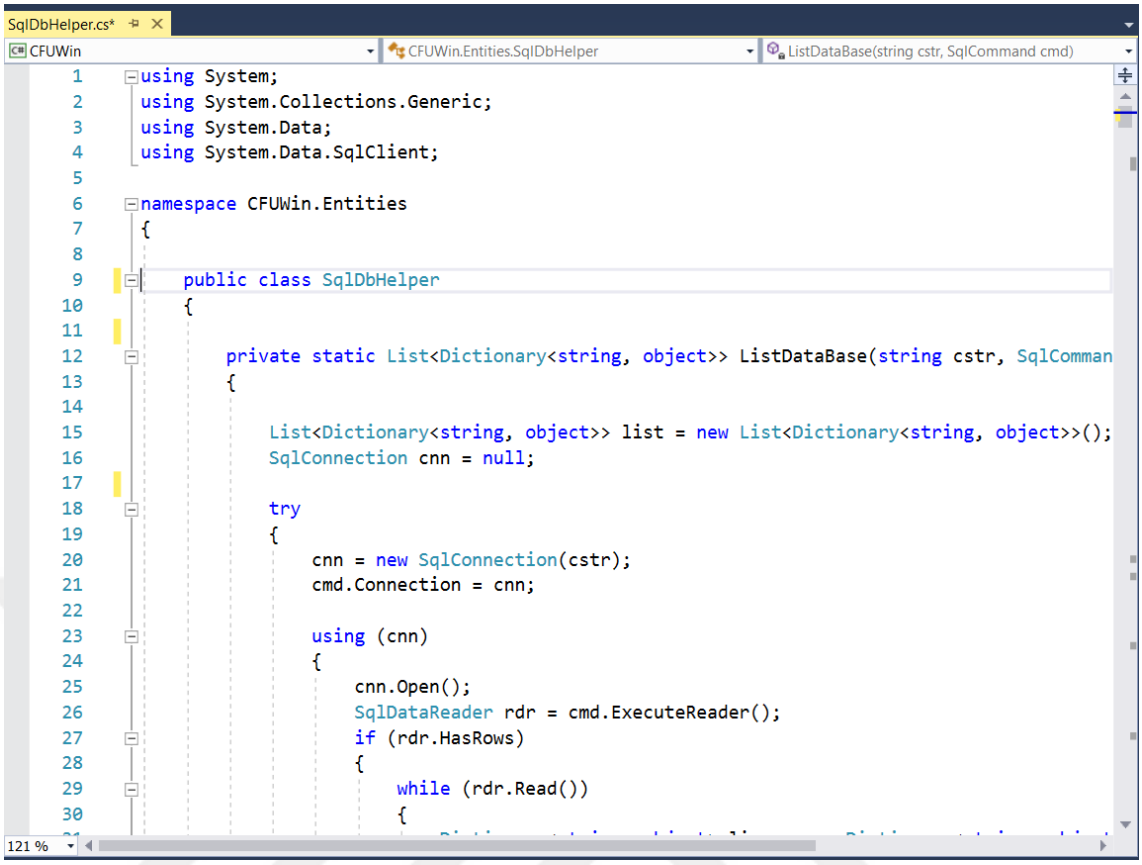
Uygulamanın kodlanmasında nesne yönelimli programlama esaslarına göre geliştirme yapılmıştır. Projenin dosya yapısında “Entities”, “Forms” ve “User Controls” Şekil 4.7.4.1. de görüldüğü üzere ayrı klasörler altında yapılandırılmıştır.



Şekil 4.7.4.1. Proje Dosyaları Klasör Yapısı

“Entites” başlığı altında tablo karşılığı varlıklar ve yardımcı nesneler kodlanmıştır. “Forms” başlığı altında kullanıcı ekranları yer alırken, “User Controls” başlığında kullanıcı arayüz bileşenleri oluşturulmuştur.

Şekil 4.7.4.2 de “Entities” klasörü altındaki “SqlDBHelper.cs” dosyası görülmektedir. Bu nesne veri erişimini ve operasyonlarını kolaylaştırmak amacıyla kodlanmış yardımcı bir sınıftır. İçindeki fonksiyonlar aracılığıyla diğer varlıklardan ve arayüzlerden veri işlemlerini gerçekleştirmek mümkündür.

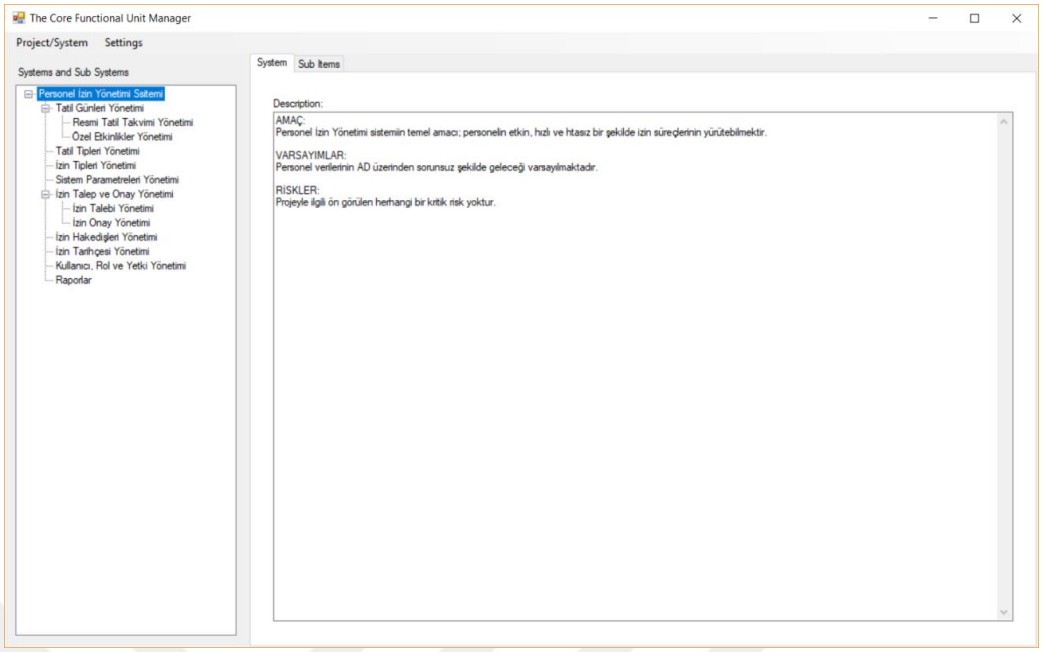


Şekil 4.7.4.2. Visual Studio Kod Penceresi

4.7.5. Örnek Arayüzler

Yazılımın ana sayfası aşağıdaki Şekil 4.7.5.1. de görüleceği üzere 3 ana bölümden oluşmaktadır:

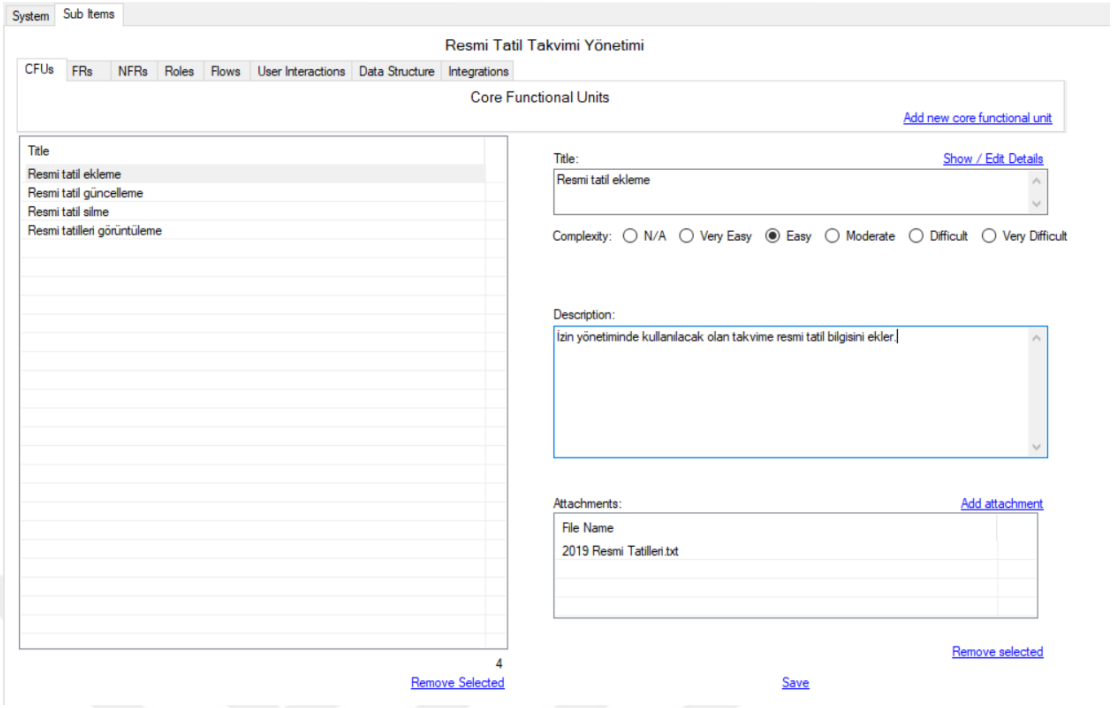
1. Menü,
2. Sistemler ağacı,
3. Sistem detay sekmeleri.



Şekil 4.7.5.1. Yazılımın Ana Ekranı

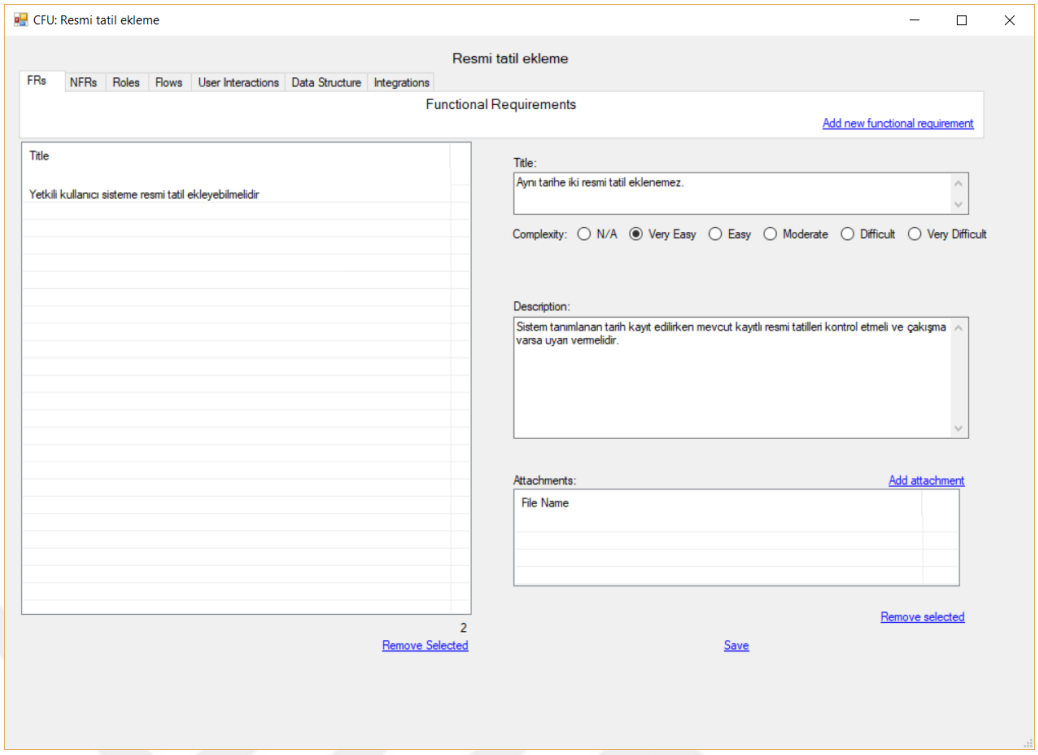
Sistem ağacında en üst seviyede proje olmak üzere alt sistemler hiyerarşisi tanımlanabilmekte ve yönetilebilmektedir. Seçilen sistem öğesi için sağ taraftaki sistem detay sekmeleri güncellenmekte ve ilgili sistemin bilgileri görüntülenmektedir.

Örneğin Şekil 4.7.5.2. deki ekran görüntüsünde “Resmî Tatil Takvimi Yönetimi” alt sisteminin İÖB listesi görüntülenmektedir.



Şekil 4.7.5.2. Alt Öğeler Ekranı

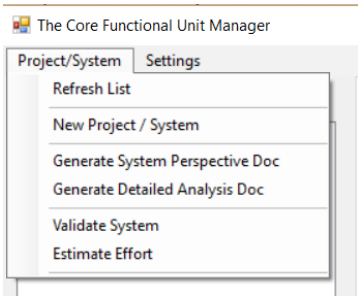
Ayrıca her bir İÖB için sol taraftaki panelden ilgili İÖB'yi seçerek sağ taraftaki "Show/Edit Details" bağlantısına tıklayarak, o İÖB için ilişkili öğelerin tanımlandığı Şekil 4.7.5.3. de görülen ekran açılabilmektedir.



Şekil 4.7.5.3. İÖB Detay Ekranı

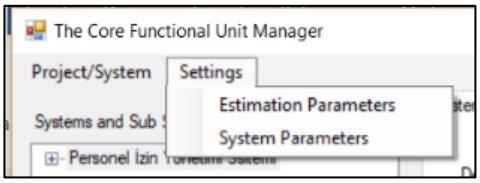
Şekil 4.7.5.3. de "Resmî Tatil Ekleme" İÖB'i ile ilişkilendirilmiş işlevsel gereksinimler görülmektedir. Diğer sekmelerde varsa diğer ilişkiler görüntülenebilmekte, yeni ilişkiler tanımlanabilmekte ya da mevcut ilişkiler silinebilmektedir.

"Sub Items" sekmelerinin her biri ilgili sistem ya da alt sistem seviyesindeki öğeleri yönetmeyi sağlamaktadır. Bu sekmelerden sistem için yeni bir İÖB ya da işlevsel gereksinim tanımlanabilmekte, mevcut bir öğe silinebilmekte ya da tanımlı öğeler listelenebilmektedir.



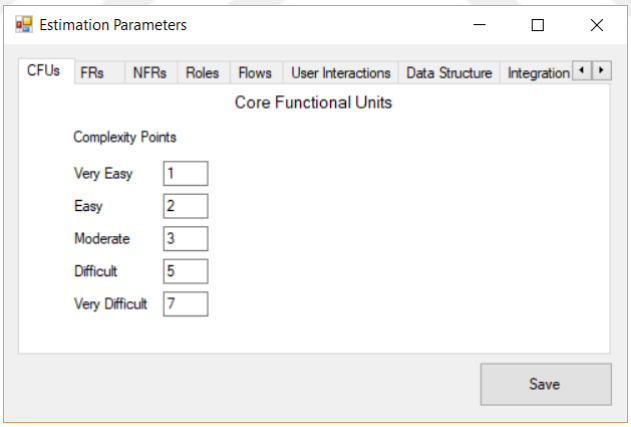
Şekil 4.7.5.4. Ana Ekran Proje / Sistem Menüsü

Yazılımın Sistem/Proje menüsünde Şekil 4.7.5.4. de görüldüğü üzere listeyi güncelleme, yeni bir proje/sistem oluşturma, sistem perspektifi yazdırma, sistem detay analiz dokümanı yazdırma, sistem doğrulaması yapma ve emek tahmini yapma seçenekleri yer almaktadır.



Şekil 4.7.5.5. Ana Ekran Ayarlar Menüsü

Ayarlar menüsündeyse Şekil 4.7.5.5. de görülen tahminleme parametreleri ve sistem parametreleri seçenekleri yer almaktadır. Şekil 4.7.5.6. de tahminleme parametreleri yönetim ekranı görülmektedir.



Şekil 4.7.5.6. Tahminleme Parametreleri Yönetim Ekranı

4.7.6. Yazılımın Kullanımı

Yazılım ilk olarak kullanılacağı zaman sistem ve tahminleme parametrelerinin gözden geçirilerek gerekiyorsa güncellenmesinde fayda bulunmaktadır.

Yeni bir projenin analizine başlamak için öncelikle "System/Project" menüsünden "New Project/System" seçeneğine tıklanır. Açılan metin kutusuna projenin adı yazılır. Böylece yeni bir sistem ve aynı zamanda proje başlatılmış olur.

Proje açıldıktan sonra analiz sürecine aşağıdaki adımlar uygulanarak yazılım etkili bir şekilde kullanılabilecektir.

Analizin ilk aşaması sistem ağacının oluşturulmasıdır. Bunun için aşağıdaki adımlar gerçekleştirilir:

1. En üst sistem seviyesi için roller, akışlar, entegrasyon ve veri yapısı tanımlanır. İstendiği taktirde bu öğeler, rol hariç, alt seviyelere de tanımlanabilmektedir,
2. Varsa projenin alt sistemleri ve onların alt sistemleri tanımlanır.

Sistem ağacı oluştuktan sonra her bir sistem ya da alt sistem için şu işlemler gerçekleştirilir:

1. Mutlaka en az bir İÖB tanımlanır. Her sistem hiyerarşik yapısı içinde en az bir tane İÖB'ye sahip olmak zorundadır,
2. Diğer bir zorunlu öğe işlevsel fonksiyondur. Her İÖB mutlaka bir işlevsel gereksinimi yerine getirmek için yazıldığından, sistem seviyesine bir ya da daha fazla işlevsel gereksinim tanımlanması zorunludur,
3. İlk iki maddeden sonra ilgili sistem ya da alt sistem için paydaşlar tarafından istenmesi durumunda kullanıcı etkileşimi, veri yapısı, İOG ve akış öğeleri tanılanabilir.

Sistem / alt sistem seviyesindeki öğelerin tanımlanması bittikten sonra her bir İÖB öğesinin detaylandırılması gerçekleştirilir. Bunun için ilgili İÖB'nin detay ekranına girilir ve ilişkili olduğu öğeler İÖB ye eklenir. Böylece analiz tamamlanmış olur.

Analist, yazılımın ilk başında ana hatlarını tamamladıktan sonra ya da ara aşamalarda dilediği zaman ana menüden sistem doğrulama fonksiyonunu çalıştırarak eksik tanımlanan öğeleri ya da zorunlu olmasına rağmen tanımlanmamış ilişkileri görebilir. Böylece analiz hatalarının azalması mümkün olacaktır.

BÖLÜM 5

MODELİN ÖRNEK UYGULAMARI

Bu bölümde ortaya koyulan gereksinim belirtimi ve analiz modeline yönelik örnek uygulama çalışmalarına değinilmektedir. Çalışma boyunca ilk ikisi modelin ilk versiyonu kullanılmak üzere üç adet örnek uygulama çalışması yapılmıştır.

İlk iki uygulama gerçek yaşam projeleri üzerinde modelin kavramsal tasarımı ile manuel olarak gerçekleştirilmiştir. Saha çalışmaları sonrasında alınan geri bildirimler ve yapılan değerlendirmeler sonrasında modelde çeşitli değişikliklere gidilmiş, model öğelerinin sayısı azaltılarak daha yalın hale getirilmiş ve modelin uygulamasının kolaylaşması için bir yazılım geliştirilmiştir. Ardından geliştirilen bu yazılımla birlikte modelin revize edilmiş hali üçüncü bir proje üzerinde kullanılmıştır. Ayrıca modelin güçlü ve zayıf yönlerinin daha iyi anlaşılabilmesi ve tartışılabilmesi için analiz biri kullanıcı hikayesi diğeri de geleneksel yöntem olmak üzere iki farklı şekilde daha gerçekleştirilmiştir.

Aşağıdaki başlıklarda sırasıyla örnek uygulamalarla ilgili senaryo ve diğer bilgiler, geliştirilen yazılım, uygulamanın analiz süreci ve bu süreç sonunda ortaya konulan çıktılar detaylarıyla açıklanmaktadır.

5.1. Personel İş Avansı ve Masraf Bildirim Yönetimi Sistemi Uygulaması

Modelin birinci versiyonuyla gerçekleştirilen ilk uygulama "Personel İş Avansı ve Masraf Bildirimi Yönetimi Sistemidir". İş kolu, sektörü ve büyüklüğü ne olursa olsun tüm kurumsal organizasyonlarda çalışanlar kurumları adına iş yaparken seyahat, konaklama, malzeme alımı gibi harcamalarda bulunmaktadırlar. Bazı kurumlar bu

harcamalar yapılmadan önce personeline iş avansı vermekte ve yapılan masraflar daha sonra bu iş avansından mahsup edilerek çalışanla kurum arasında mahsuplaşma yapılmaktadır. Örnek proje seçilirken tüm kurumlarda bilinen ve işletilen bir süreç olması en önemli kriter olmuştur.

5.1.1. Örnek Uygulamanın Ön Analizi

Örnek uygulamanın analiz sürecinde öncelikle iş biriminden bir ürün sahibi belirlenmiş ve yazılım geliştirme birimine proje talebi iletilmiştir. Bu noktada yazılım geliştirme birimince projeye atanan analist, iş birimindeki ürün sahibiyle birlikte anahtar paydaşların sistemle ilgili beklentilerini ve sistemden yapmak istedikleri temel öğeleri tespit etmiştir. Sponsordan alınan sistem amacı şu şekilde tarif edilmiştir:

"Personel avans taleplerinin ve masraf bildirimlerinin hızlı ve denetlenebilir şekilde işletilebilmesi."

Analist bu iş ihtiyacını ve sistem amacını aldıktan sonra, verilen bilginin gereksinimlere girerken, kaliteli bir gereksinimin sahip olması gereken temel özelliklerden biri olan atomik olma prensibini göz önünde bulundurarak sistemin amaçlarını iki alt başlığa bölmüştür:

Sistem Amaçları:

1. Personel avans taleplerinin hızlı ve denetlenebilir şekilde işletilebilmesi,
2. Personel masraf bildirimlerinin hızlı ve denetlenebilir şekilde işletilebilmesi.

Böylece ön analiz sürecinin ilk adımı olan "sistem amaçlarının belirlenmesi" tamamlanmış olmaktadır.

Sistemin amacı ya da amaçları belirlendikten sonra ön analiz sürecinde tanımlanmış olan "temel iş akışlarını belirleme" adımına geçilmiştir. Bu aşama ürün sahibi ve sponsorla birlikte sistemdeki en kritik ve temel iş adımları üzerine beyin fırtınası yapılmış ve temel iş akışları olarak ana adımlarıyla birlikte aşağıdaki akışlar belirlenmiştir:

Sistemdeki Temel İş Akışları:

1. Avans Bildirimi: avans talebi, yönetici onayı, mali işler onayı, ödeme,
2. Masraf Bildirimi: masraf bildirimi, yönetici onayı, mali işler kontrolü, mal işler onayı, ödeme.

Temel iş akışları da belirlendikten sonra sistem amaçları ve akışlar göz önünde bulundurularak sistemi kullanacak kullanıcılar, kullanıcı gruplar ve dolayısıyla rollerin tespiti için çalışma yapılmıştır. Bu çalışma sonrasında sistemi kullanacak roller şu şekilde belirlenmiştir:

Roller:

1. Personel, tüm çalışanlar…
2. Birim Yöneticisi, masraf ve avans bildirimi yapan personelin bağlı olduğu yönetici,
3. Mali İşler Uzmanı,
4. Mali İşler Yöneticisi,
5. Sistem Yöneticisi.

Kullanıcı rollerinin belirlenmesinden sonra sırasıyla dış sistem entegrasyonları, alt sistem yapısı çalışılmış ve sistem için şu bilgiler ortaya konulmuştur:

Entegrasyonlar:

1. Active Directory, kullanıcı bilgileri ve bağlı oldukları yönetici bilgileri.

Alt Sistemler:

1. Avans Talep Yönetimi,
2. Masraf Bildirim Yönetimi,
3. Proje Kataloğu Yönetimi,
4. Masraf Limitleri Yönetimi.

Son olarak tüm bu çalışmalar boyunca aralarda telaffuz edilen işlevsel olmayan gereksinimler belirlenerek yazılı hale getirilmiştir.

1. Sistem web uygulaması olarak çalışmalıdır,

2. Tüm sayfalar SSL üzerinde çalışmalıdır,

3. Sistemde ön tanımlı limitler dâhilinde otomatik işlemler yapılabilmelidir.

Ön analiz sürecinde yukarıdaki adımlar tamamlandıktan sonra elde edilen bilgiler Şekil 5.1.1.1. de gösterilen "sistem perspektifi" dokümanına yazılmış ve kilit paydaşlardan (ürün sahibi ve sponsor) teyit istenmiştir. Kilit paydaşlarca belirtilen birkaç düzeltmeden sonra ürün perspektifi onaylanmış ve ön analiz aşaması tamamlanmıştır.

Personel İş Avansı Talebi ve Masraf Bildirim Yönetimi – Sistem Perspektifi			
Amaçlar:	1. Personel avans taleplerinin hızlı ve denetlenebilir şekilde işletilebilmesi. 2. Personel masraf bildirimlerinin hızlı ve denetlenebilir şekilde işletilebilmesi	**Roller:**	1. Personel, tüm çalışanlar… 2. Birim Yöneticisi, masraf ve avans bildirimi yapan personelin bağlı olduğu yönetici. 3. Mali İşler Uzmanı 4. Mali İşler Yöneticisi 5. Sistem Yöneticisi
İş Akışları	**İşlevsel Olmayan Gereksinimler**	**Alt Sistemler ve Modüller**	**Dış Sistemler ve Entegrasyon**
1. Avans Bildirimi: avans talebi, yönetici onayı, mali işler onayı ödeme. 2. Masraf Bildirimi: masraf bildirimi, yönetici onayı, mali işler kontrolü, mali işler onayı, ödeme	1. Sistem web uygulaması olarak çalışacaktır. 2. Tüm sayfalar SSL üzerinde çalışmalıdır. 3. Sistemde ön tanımlı limitler dahilinde otomatik işlemler yapılabilmelidir.	1. Avans Talep Yönetimi 2. Masraf Bildirim Yönetimi 3. Proje Kataloğu Yönetimi 4. Masraf Limitleri Yönetimi	1. Active Directory, kullanıcı bilgileri ve bağlı oldukları yönetici bilgileri
NOTLAR:			
Hazırlayan:		**Onaylayan Paydaşlar**	
Tarih:		**Adı/Unvanı/Görevi**	**Tarih**
Versiyon:			

Şekil 5.1.1.1. Sistem Perspektif Dokümanı

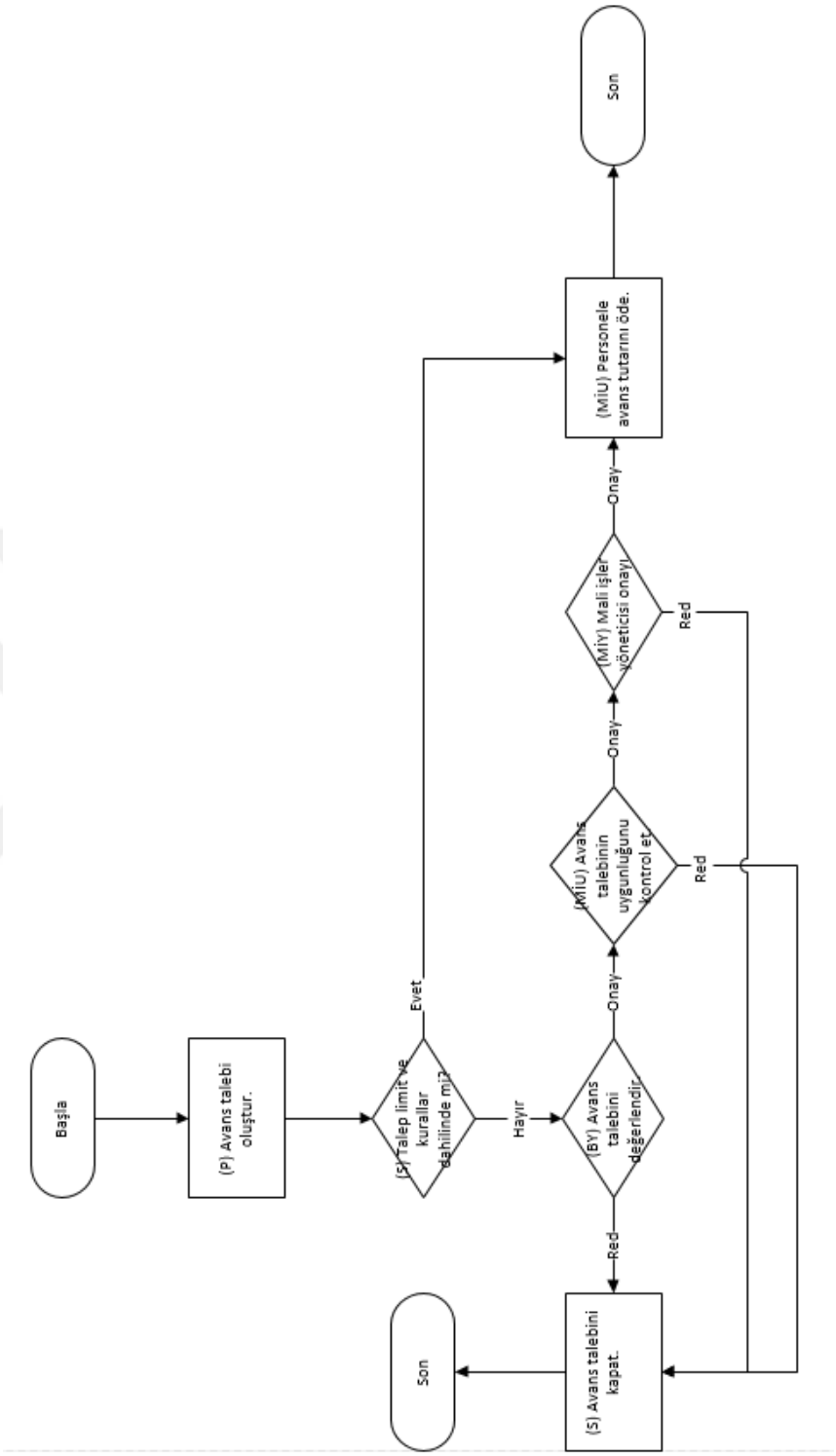
5.1.2. Örnek Uygulamanın Detaylı Analizi

Örnek uygulamanın detay analizinde modelin önerdiği doğrultuda sistem perspektifi dokümanı esas alınarak yinelemeli ve döngüsel olarak sistem analizi – alt sistem analizi süreci işletilmiştir. Bu aşamada öncelikle sistemin temel öğeleri detaylandırılmış, ancak durum gerektirdiğinde alt sistem detayına da girilmiş ve böylece kademeli olarak her seferinde biraz daha ayrıntının ortaya çıktığı bir analiz süreci gerçekleştirilmiştir.

Modelin sistem analizi aşaması çalıştırıldığında ilk olarak amaçlarla ilgili bir güncelleme olup olmadığının belirlenmesi adımı işletilmiştir. Bu aşamada amaçların yeterince açık, net ve eksiksiz olduğu belirlenmiştir.

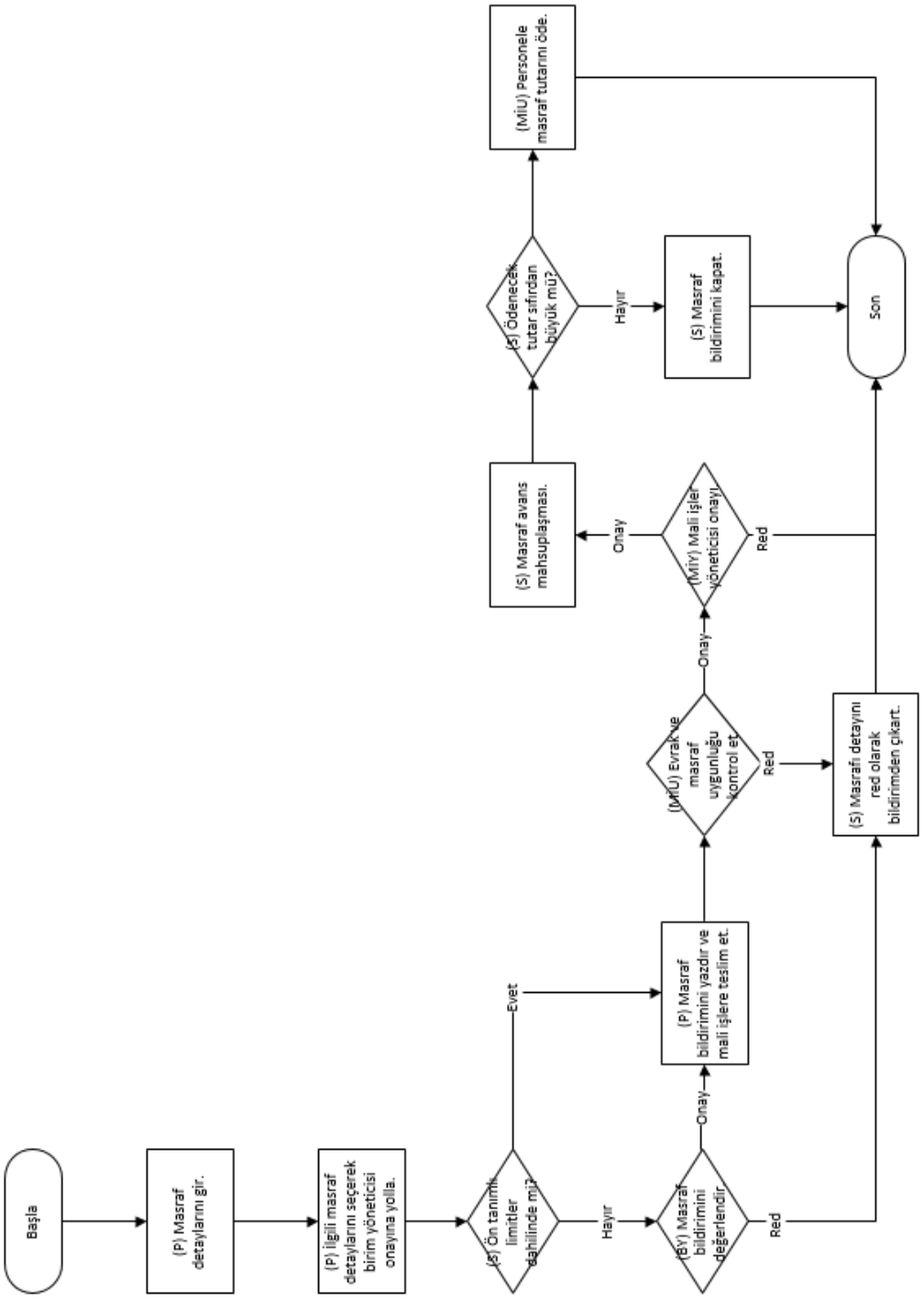
Ardından iş akışları ele alınmıştır. Burada ön analiz aşamasında belirlenen 2 temel iş akışı olan “avans bildirim” ve “masraf bildirimi” iş akışlarıyla başlanmış ve başka iş akışı olup olmadığı üzerine çalışılmıştır. Çalışma sonunda başka bir iş akışı bulunmadığı konusu netleştirilmiş ve mevcut iki iş akışı detaylandırılmıştır.

Şekil 5.1.2.1. de avans bildirimi akış diyagramını görmektesiniz görülmektedir.



Şekil 5.1.2.1. Avans Bildirimi Akış Diyagramı

Şekil 5.1.2.2. de ise masraf bildirimi akışı gösterilmektedir.



Şekil 5.1.2.2. Masraf Bildirimi Akış Diyagramı

Her iki diyagramda da görülebileceği üzere, iş kuralları ve adımlar iş birimince tarif edildiği şekilde yazılmıştır. Ek olarak hangi görevin/adımın hangi rol tarafından gerçekleştirileceği eklenmiştir. Bu aşamada çıktı olarak ortaya konulan doküman iş akışları dokümanıdır.

Akış diyagramları ilk oluşturulurken iş biriminin anlattığı adımlarla çizilir. Ancak detay analiz aşamasında ilerlendikçe her bir adım işlevsel öz birimlerden oluşacak şekilde güncellenirler.

Akış diyagramları belirli bir olgunluk seviyesine ulaştıktan sonra arayüz tasarımı konuşulmaya başlanır. Bu aşamada iş akışlarının veri girişi, onay vb. etkileşimi ve kullanıcıya bilgi aktarımı için hangi ekranların ve diyalogların olması gerektiği belirlenir. Arayüzler belirlenirken öncelikle ekranların isimleri, daha sonra alanları ve daha sonra da sayfa düzeni, ekran kontrol tipleri ve kullanıcı veri girişi doğrulama bilgileri detaylandırılır.

Örnek uygulamanın arayüz analizinde belirlenen ekranların ya da ekran öğelerinin listesi şöyledir:

1. Masraf bildirimleri listesi,
2. Taslak Masraf bildirim detayı,
3. Masraf bildirim detayı,
4. Masraf detayı ekleme,
5. Masraf detayı güncelleme,
6. Masraf detayı silme,
7. Birim yöneticisi onay bekleyen masraf bildirimleri listesi,
8. Birim yöneticisi masraf onay/red,
9. Kontrol bekleyen masraf bildirimleri listesi,
10. Mali işler uzmanı masraf detayı onay/red,
11. Mali işler yöneticisi onay bekleyen masraf bildirimleri listesi,
12. Mali işler yöneticisi bildirim onay/red,
13. Ödenecek masraf bildirimleri listesi,

14. Masraf ödeme kaydı girişi,
15. Avans geçmişi listesi görüntüleme,
16. Avans talebi görüntüleme,
17. Avans talep girişi,
18. Avans talep güncelleme,
19. Avans talebi iptal etme,
20. Birim yöneticisi onay bekleyen avans taleplerini listesi,
21. Birim yöneticisi avans talebi onay/red,
22. Kontrol bekleyen avans taleplerini listele
23. Mali işler uzmanı avans talebi onay/red,
24. Mali işler yöneticisi onay bekleyen avans talepleri listesi,
25. Mali işler yöneticisi avans talebi onay/red,
26. Ödenecek avans talepleri listesi,
27. Ödeme kaydı girişi,
28. Proje listesi,
29. Masraf limitleri görüntüleme,
30. Masraf limitleri güncelleme,
31. Masraf kalemleri listesi,
32. Yeni masraf kalemi tanımlama,
33. Masraf kalemi bilgileri güncelle,
34. Performans limitleri görüntüleme,
35. Performans limitleri güncelleme,
36. Sisteme giriş,
37. Kullanıcı ana sayfası
38. A/D- sistem rol eşleştirme,

39. Kişi bazında işlem logu görüntüleme,

40. İşlem bazında log detayı görüntüleme,

41. Durum aksiyon özetini görüntüleme,

42. Sistem ayarları.

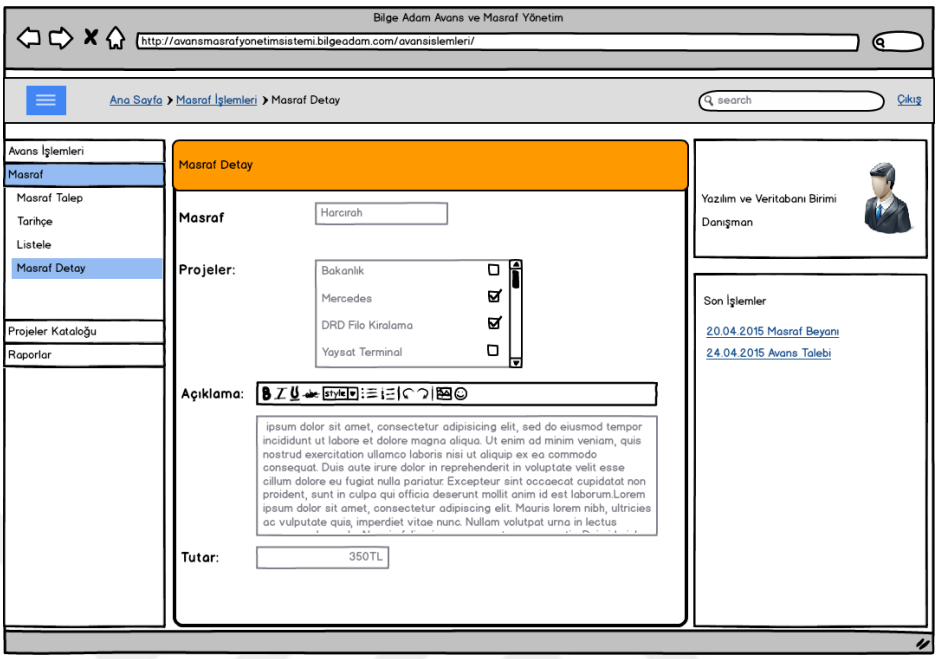
Uygulama sırasında arayüz isimleri detay analiz sürecinde bir kerede belirlenemeyebilir. Kademeli detaylandırma yaklaşımında iş akışları, alt sistemler, veri yapısı detaylandıkça ekranlar ve ekran detayları da aşamalı olarak ortaya çıkacaktır. Arayüzlerin analiziyle birlikte ortaya ekran analizleri dokümanı çıkar. Bu dokümanın ilk aşamalarında ekran isimleri ve ekranlarla ilgili özet açıklamalar yer alırken ilerleyen aşamalarında ekranlarla ilgili prototip çizimleri ve kontrol bazında detay yer alabilir.

Aşağıda örnek uygulamayla ilgili ekranlardan bazılarını ve özet açıklamalar görülmektedir.

Çizelge 5.1.2.1. Sistem Perspektif Dokümanı

ID	**Ekran Adı**	**Açıklama**
1	Masraf bildirimleri listesi	Taslak masraf bildirimi ve diğer masraf bildirimleri listesidir. Masraf no, bildirim tarihi, durumu, tutarı gibi bilgiler yer alır. Tarihe göre ve bildirim durumuna göre sıralanır.
2	Taslak Masraf bildirim detayı	Her bir masraf bildiriminin içine girildiğinde buradaki masraf detayları (fiş, fatura, harcama detayı) listelenir. Ayrıca kullanıcı burada istediği masraf detaylarını seçerek yönetim onayına yeni bir masraf bildirimi olarak yollar. Birim yöneticisine ya da üst yönetime onaya yollayabilir.
3	Masraf bildirim detayı	Taslak olmayan masraf bildirimleri için bildirim detayıdır
4	Masraf detayı ekleme	Yeni bir masraf girişi yapmak için kullanılır. Masraf tarihi, açıklama, harcama kalemi, ilgili projeler, vb. bilgiler yer alır

Örnek uygulamayla ilgili bir başka ekran prototipi Şekil 5.1.2.3. de görülmektedir.



Şekil 5.1.2.3. Masraf Detay Ekranı

Detaylı analiz aşamasında sistemin veri modeli ve veri yapısı ortaya çıkmaya başlamaktadır. Bu aşamada yapılan kademeli derinleştirmede ilk olarak masraf, avans, kullanıcı, proje gibi temel sistem öğeleri tespit edilmiştir. Ardından sistem ve alt sistem analizi aşamalarında kademeli olarak detaylandırma yapıldıkça öğeler çeşitlenmiş ve son olarak aşağıdaki veri öğeleri tespit edilmiştir.

1. Masraf bildirimi,
2. Avans talebi,
3. Kullanıcı,
4. Rol eşleşmeleri,
5. Sistem rolleri,
6. İşlemler,
7. Konfigürasyon,

8. Müşteri,
9. Proje,
10. Masraf kalemi,
11. Masraf detayı.

Her bir veri modeli ve detayı zamanla olgunlaştırılarak her biri için aşağıdaki tablo oluşturulmuş ve veri modeli ve detayı dokümanı oluşturulmuştur.

Çizelge 5.1.2.2. Veri Modeli ve Detayı Dokümanı

Attribute	Zorun	Kaynak	Veri Tipi	Açıklama	iş Kuralları
Masraf Bildirim No	x	sistem	Tam sayı		Üretilen her bildirim için sistem tarafından otomatik olarak ardışık bir numara verilir.
Beyan Tarihi	x	sistem	Tarih		
Son Güncellenme Tarihi		sistem	Tarih		
Toplam Tutar		sistem	Para		hesaplama
Belge Sayısı		sistem	tamsayı		hesaplama
Masraf Detayları		kullanıcı	Masraf Detayı		
İşlem Durumu	x	sistem	Durum		
BY Onayına Yollanma Tarihi		sistem	Tarih		
BY Onay Tarihi		sistem	Tarih	Tüm detayların onaylanmasının tamamlandığı tarihtir.	
MİU Onayına Yollanma Tarihi		sistem	Tarih		
MİU Onay Tarihi		sistem	Tarih	Tüm detayların onaylanmasının tamamlandığı tarihtir.	
MİY Onayına Yollanma Tarihi		sistem	Tarih		
MİY Onay Durumu		sistem	Tamsayı	işlem yapılmadı, onaylandı, reddedildi	
MİY Açıklaması		kullanıcı	Metin		
İşlem Yapan MİY		sistem	Kullanıcı		
MİYin Belirlediği Ödeme Tarihi		kullanıcı	Tarih		
Ödenme Tarihi		sistem	Tarih		
Ödemeyi Yapan		sistem	Kullanıcı		
Ödeme Açıklaması		kullanıcı	Metin		
Otomatik Mahsuplaşılan Avanslar		sistem	Avans (liste)	masraf ödeme durumuna gelirken eğer personelin üzerinde kapatılmamış avans varsa öncelikle sistem tarafından mahsuplaşma yapılır	
Mahsup edilen toplam tutar		sistem	para		

Bu çalışmalar esnasında yapılan detay analizin her aşamasından gelen İOG bildirimleri "İşlevsel Olmayan Gereksinimler" başlığı altına yazılmıştır. Örnek uygulama için tespit edilen 4 adet işlevsel olmayan gereksinim Çizelge 5.1.2.3. deki gibidir:

Çizelge 5.1.2.3. İşlevsel Olmayan Gereksinimler

SN	İşlevsel Olmayan Özellikler
1	Sistem temel olarak web uygulama olarak geliştirilecektir.
2	Web uygulama SSL üzerinde çalışacaktır.
3	Yazılım C# 5 programlama diliyle ve ASP.NET MVC 5 teknolojisiyle geliştirilmelidir.
4	Veritabanı olarak MS SQL Server 2012 kullanılmalıdır.

Bu aşamadan sonra detay analizin sistem analiz aşaması tamamlanmış olur ve alt sistem analizi aşamasına geçilir. Gerçekleştirilen çalışmada alt sistem analizi aşamasında belirlenen İÖB'ler 49 adet olarak aşağıda listelenmiştir.

1. Masraf bildirimlerini listele,
2. Masraf bildirimi detaylarını getir,
3. Masraf detayı ekle,
4. Masraf detayını güncelle,
5. Masraf detayını sil,
6. Masraf beyanını BY onayına yolla,
7. Masraf bildirimini yazdır,
8. Ön tanımlı kurallardan otomatik onaylı bildirim oluştur,
9. Bağlı personelin onay bekleyen bildirimlerini listele,
10. Masraf detayını onayla/reddet,
11. Tüm personelin kontrol bekleyen bildirimlerini listele,
12. Masraf detayını onayla/reddet,
13. Tüm personelin kontrol bekleyen bildirimlerini listele,
14. Masraf bildirimini onayla/reddet,
15. Otomatik masraf/avans mahsup et,

16. Ödenecek masraf bildirimlerini listele,

17. Ödeme girişi yap,

18. Avans taleplerini listele,

19. Avans talebi bilgilerini getir,

20. Yeni avans talebi oluştur,

21. Avans talebini iptal et,

22. Ön tanımlı kurallardan otomatik onaylama yap,

23. Bağlı personelin onay bekleyen avans taleplerini listele,

24. Avans talebini onayla/reddet,

25. Tüm personelin kontrol bekleyen avans taleplerini listele,

26. Avans talebini onayla/reddet,

27. Tüm personelin kontrol bekleyen avans taleplerini listele,

28. Avans talebini onayla/reddet,

29. Ödenecek avans taleplerini listele,

30. Ödeme girişi yap,

31. Avans talebi güncelle,

32. Projeleri listele,

33. Proje kapat,

34. Masraf limitlerini görüntüle,

35. Masraf limitlerini güncelle,

36. Masraf kalemlerini listele,

37. Yeni masraf kalemi ekle,

38. Mevcut masraf kalemi bilgilerini güncelle,

39. Masraf kalemi kapat,

40. Performans (SLA) bilgilerini görüntüle,

41. Performans (SLA) bilgilerini güncelle,

42. Sisteme giriş yap,

43. Sistemden güvenli çıkış,

44. Active Directory rolleriyle sistem rolleri eşleştirme,

45. Kişi bazında işlem logu görüntüle,

46. İşlem bazında log detayı görüntüle,

47. Durum aksiyon özetini görüntüle,

48. Durum değiştirme ve bildirim yollama,

49. Sistem ayarlarını güncelle.

Detay analiz dokümanında işlevsel öz birimler sistem – alt sistem hiyerarşisi içinde tanımlanmıştır. Bu yapı içerisinde her bir işlevsel öz birim bir başlık olarak açılmış ve bu başlık altında işlevsel öz birimle ilgili diğer öğeler ve ilişkileri detaylı olarak açıklanmıştır.

Çizelge 5.1.2.4. deki örnekte birim yöneticisinin kendine bağlı personelin avans taleplerini listelemesine ilişkin işlevsel öz birim başlığı ve detayları görülmektedir.

Çizelge 5.1.2.4. İşlevsel Öz Birim Kartı

Başlık	**Açıklama**
İşlevsel Öz Birim	Bağlı personelin onay bekleyen avans taleplerini listele
Açıklama	Birim yöneticisinin kendine bağlı personelin onay bekleyen avans taleplerini listelemesi.
Ekran	Birim yöneticisi Onay bekleyen avans taleplerini listeleme ekranı
İş Kuralı	(Yok)
İşlem Tipi	Veri okuma
Veri Modeli	Avans talebi, kullanıcı, proje

Veri Yapısı	Avans talep no, avans talep tarihi, avans tutarı, ilişkili projeler, açıklama
İşlevsel Olmayan Gereksinimler	(yok)
İş Akışı / Senaryo	Avans bildirimi
Kullanıcı Rolü	Birim yöneticisi
Alt sistem / modül	Avans Alt Sistemi

5.1.3. Örnek Uygulamanın Karmaşıklık ve Büyüklük Tahminlemesi ve Gerçekleşme

Örnek uygulama için belirlenen 49 işlevsel öz birim için toplam 338,4 puan hesaplanmıştır. Bu puan için belirlenen puan / adam-saat çarpanı her bir puan için 1 saat olarak tahmin edilmiş ve sonuç olarak işin 338,4 saatte yani 42,25 günde bitirileceği ön görülmüştür. Örnek uygulama için çalışan yazılımcıya önerilen yeni modelden bahsedilmeden analizle ilgili gereksinimler ve ekran prototipleri gösterilerek bir tahmin yapması istenmiş, yazılımcıdan sistemle ilgili 30 günlük bir tahmin alınmıştır. Proje ekibinin diğer görevleri nedeniyle yazılım 6 aylık bir takvim süreci içerisinde 37 günde bitirmiştir.

Puanlamanın doğru olduğu varsayımı üzerinden harcanan proje emeği günlük 8 saat hesaplanarak saate çevrildiğinde ortaya 296 saat çıkmaktadır. Bu süre toplam puan olan 338'e bölündüğünde bir puanın gerçekleşmesi için 0,87 saat yani yaklaşık 52 dakika gerektiği bulunmuştur.

Aşağıda örnek projenin emek hesaplamasında kullanılan tablonun bir kısmı gösterilmektedir.

Çizelge 5.1.3.1. Emek Hesaplama Tablosu

		Karmaşıklık Puanı						
SN	İşlevsel Öz Birim	Ara Yüz	İş Kuralı	Veritabanı Opr	Veri Modeli ve Yapısı	Baz Puan	İşlevsel Olmayan Gereksinim Çarpanı	Toplam Puan
31	Avans talebi güncelle	2	2	1	2	7	1.1	7.7
32	Projeleri listele	2	1	1	1	5	1.1	5.5
33	Proje kapat	2	2	1	1	6	1.1	6.6
34	Masraf limitlerini görüntüle	3	0	3	1	7	1.1	7.7
35	Masraf limitlerini güncelle	3	1	2	2	8	1.1	8.8
36	Masraf kalemlerini listele	2	0	1	1	4	1.1	4.4
37	Yeni masraf kalemi ekle	2	2	1	2	7	1.1	7.7
38	Mevcut masraf kalemi bilgilerini güncelle	2	2	1	2	7	1.1	7.7
39	Masraf kalemi kapat	1	1	1	1	4	1.1	4.4
40	Performans (SLA) bilgilerini görüntüle	2	0	3	1	6	1.1	6.6
41	Performans (SLA) bilgilerini güncelle	2	2	2	3	9	1.1	9.9
42	Sisteme giriş	1	2	1	2	6	1.1	6.6
43	Sistemden güvenli çıkış	1	1	1	1	4	1.1	4.4
44	Active Directory rolleriyle sistem rolleri eşleştirme	0	1	2	2	5	1.1	5.5
45	Kişi bazında işlem logu görüntüle	3	1	2	1	7	1.1	7.7
46	İşlem bazında log detayı görüntüle	3	1	2	1	7	1.1	7.7
47	Durum aksiyon özetini görüntüle	3	1	2	1	7	1.1	7.7
48	Durum değiştirme ve bildirim yollama	3	1	2	1	7	1.1	7.7
49	Sistem ayarlarını güncelle.	3	2	2	2	9	1.1	9.9
							Puan	338.4

Yapılan emek tahminleme çalışmasının özet tablosu şu şekildedir:

Çizelge 5.1.3.2. Tahminleme Sonuç Özet Tablosu

Açıklama	**Değer**
Toplam Puan	338
Puan başına ön görülen emek (adam saat)	1
Proje için ön görülen toplam emek (adam saat)	338
Proje için ön görülen toplam emek (adam gün) (günlük 8 saat çalışma esas alınmıştır.)	42
Projenin gerçekleşme süresi (adam gün)	37
Tahminlenen ile gerçekleşen arasındaki sapma oranı	%13,5
Gerçekleşene göre puan başına emek (adam saat)	0,87

5.2. Personel İzin Yönetimi Uygulaması

Modelin birinci versiyonuyla gerçekleştirilen ikinci uygulama olan "Personel İzin Yönetimi" yazılımı, bir diğer örnek olarak ele alınan "Personel İş Avansı ve Masraf Yönetimi" yazılımına benzer şekilde iş kolu, sektörü ve büyüklüğü ne olursa olsun tüm kurumsal organizasyonlarda yer alan bir süreçtir.

Bu başlıkta ikinci örnek olarak seçilen "Personel İzin Yönetimi Uygulaması" ile ilgili çalışmalar aktarılmaktadır. Bir önceki bölümde çalışma biçimi ve süreç ayrıntılı olarak aktarıldığında bu bölümde yapılan uygulamalar sadece projeye odaklı olarak ele alınmıştır.

5.2.1. Örnek Uygulamanın Ön Analizi

Projeyle ilişkili paydaşlardan alınan bilgiler doğrultusunda ilk olarak sistemin amacı tespit edilmiştir.

"Personel izin sürecinin hatasız, minimum emek ile en hızlı şekilde iletmek."

Sistemin amacı belirlendikten sonra "temel iş akışlarını belirleme" adımına geçilmiştir. Bu aşama ilgili paydaşlarla sistemdeki en kritik ve temel iş adımları üzerin beyin fırtınası yapılmış ve temel iş akışları olarak ana adımlarıyla birlikte aşağıdaki sistemin tek bir iş akışından oluştuğuna karar verilmiş ve iş akışı "İzin talep yönetimi iş akışı" olarak isimlendirilmiştir.

Temel iş akışları da belirlendikten sonra sistem amaçları ve akışlar göz önünde bulundurularak sistemi kullanacak kullanıcılar, kullanıcı gruplar ve dolayısıyla rollerin tespiti için çalışma yapılmıştır. Bu çalışma sonrasında sistemi kullanacak roller şu şekilde belirlenmiştir:

Roller:

1. Personel (tüm çalışanlar),
2. Birim Yöneticisi (personelin iznini onaylamaya yetkili amiri),
3. İK Uzmanı,

4. Sistem,
5. Sistem Yöneticisi.

Kullanıcı rollerinin belirlenmesinden sonra sırasıyla dış sistem entegrasyonları, alt sistem yapısı çalışılmış ve sistem için şu bilgiler ortaya konulmuştur:

Entegrasyonlar:

1. Active Directory, kullanıcı bilgileri ve bağlı oldukları yönetici bilgileri.

Alt Sistemler:

1. İzin Talep Yönetimi,
2. İzin Bakiye Yönetimi,
3. İzin Raporları.

Son olarak tüm bu çalışmalar boyunca telaffuz edilen İOG'ler yazılı hale getirilmiştir.

1. Sistem web uygulaması olarak çalışmalıdır. Ve kurum dışından da erişilebilir olmalıdır.
2. Tüm sayfalar SSL üzerinde çalışmalıdır.

Ön analiz sürecinde yukarıdaki adımlar tamamlandıktan sonra elde edilen bilgilerle Şekil 5.2.1.1. 'dekine benzer bir "sistem perspektifi" dokümanına yazılmış ve kilit paydaşlardan (ürün sahibi ve sponsor) teyit istenmiştir. Kilit paydaşlarca belirtilen birkaç düzeltmeden sonra ürün perspektifi onaylanmış ve ön analiz aşaması tamamlanmıştır.

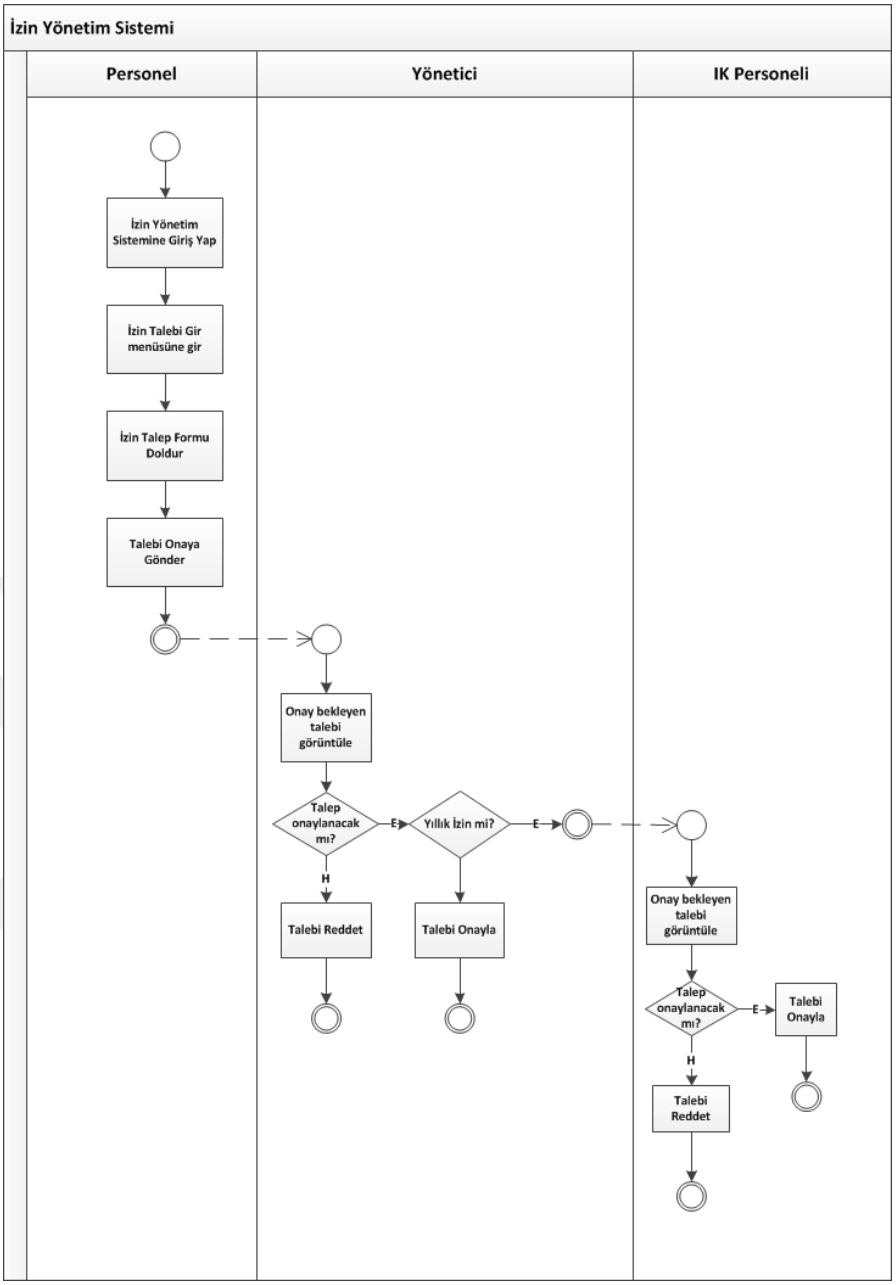
Personel İzin Yönetimi – Sistem Perspektifi			
Amaçlar: • Personel izin sürecinin hatasız, minimum eforla en hızlı şekilde iletmek."		**Roller:** • Personel (tüm çalışanlar). • Birim Yöneticisi (personelin iznini onaylamaya yetkili amiri). • İK Uzmanı • Sistem • Sistem Yöneticisi	
İş Akışları	**İşlevsel Olmayan Gereksinimler**	**Alt Sistemler ve Modüller**	**Dış Sistemler ve Entegrasyon**
• İzin talep yönetimi iş akışı	• Sistem web uygulaması olarak çalışacaktır. Ve kurum dışından da erişilebilir olacaktır. • Tüm sayfalar SSL üzerinde çalışmalıdır.	• İzin Talep Yönetimi • İzin Bakiye Yönetimi • İzin Raporları	• Active Directory, kullanıcı bilgileri ve bağlı oldukları yönetici bilgileri
NOTLAR:			
Hazırlayan:		**Onaylayan Paydaşlar**	
Tarih:		**Adı / Unvanı / Görevi**	**Tarih**
Versiyon:			

Şekil 5.2.1.1. Personel İzin Yönetimi Sistem Perspektif Dokümanı

5.2.2. Örnek Uygulamanın Detaylı Analizi

Örnek uygulamanın detay analizinde modelin önerdiği doğrultuda sistem perspektifi dokümanı esas alınarak yinelemeli ve döngüsel olarak sistem analizi süreci işletilmiştir. Bu aşamada öncelikle sistemin temel öğeleri detaylandırılmış, ancak durum gerektirdiğinde alt sistem detayını da girilmiş ve böylece kademeli olarak her seferinde biraz daha ayrıntının ortaya çıktığı bir analiz süreci gerçekleştirilmiştir.

Ardından sistemin iş akışı ele alınarak çizimi yapılmıştır. Şekil 5.2.2.1. de personel izin yönetim sistemi akış diyagramını görülmektedir.



Şekil 5.2.2.1. Personel İzin Yönetimi İş Akışı

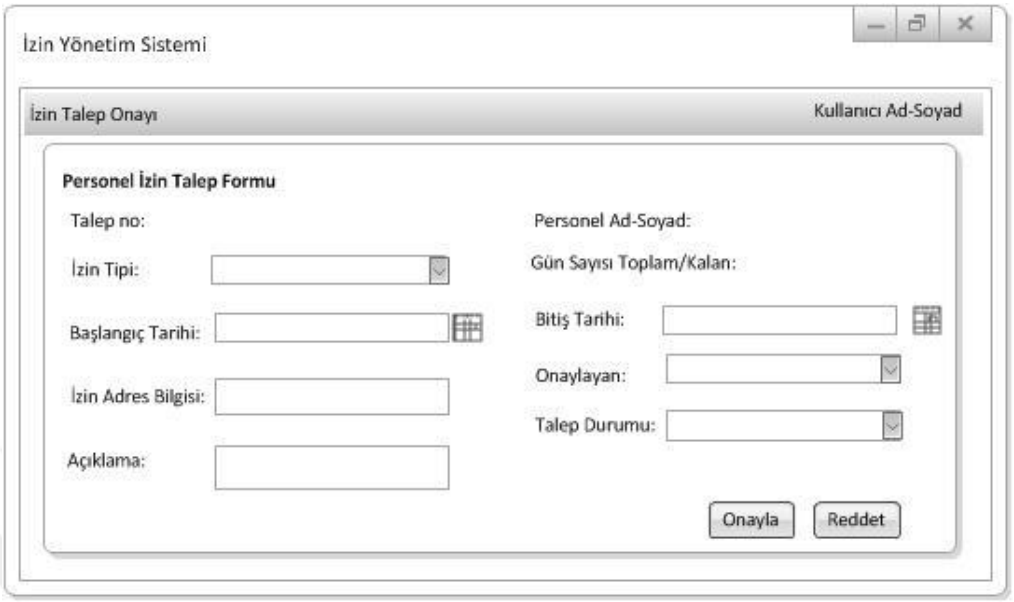
Akış diyagramı çizildikten sonra üzerinden birkaç kez geçerek alternatif durumlar tartışıldı ve İÖB'ler tespit edilmeye çalışıldı. Belirli bir olgunluk seviyesine ulaştıktan sonra arayüzler konuşulmaya başlanır. Bu aşamada iş akışlarının veri girişi, onay vb. etkileşimi ve kullanıcıya bilgi aktarımı için hangi ekranların ve diyalogların olması gerektiği belirlenir. Arayüzler belirlenirken öncelikle ekranların isimleri, daha sonra

alanları ve daha sonra da sayfa düzeni, ekran kontrol tipleri ve kullanıcı veri girişi doğrulama bilgileri detaylandırılır.

Örnek uygulamanın arayüz analizinde belirlenen ekranların ya da ekran öğelerinin listesi şöyledir:

1. Sistem Giriş Sayfası,
2. Ana Sayfa,
3. İzin Talebi Ekranı,
4. İzin Talep Onay Ekranı,
5. Talebe Onay Verme Ekranı,
6. İzin Talep Geçmişi Ekranı,
7. İzinli Personel Listesi,
8. Eskalasyon Log Listesi,
9. Parametre/SLA Güncelleme Ekranı,
10. Personel Arama/Listeleme Ekranı,
11. Personel İzin Bakiyesi Görüntüleme/Güncelleme Ekranı,
12. Geçmiş Dönem İzin Raporu,
13. Sistem Parametreleri Ekranı,
14. İşlem Logları Ekranı.

Aşağıda belirlenen ve prototipi çizilen ekran görüntülerinden bazıları verilmiştir. Şekil 5.2.2.2. de "İzin Talep Onay Ekranı" görülmektedi.

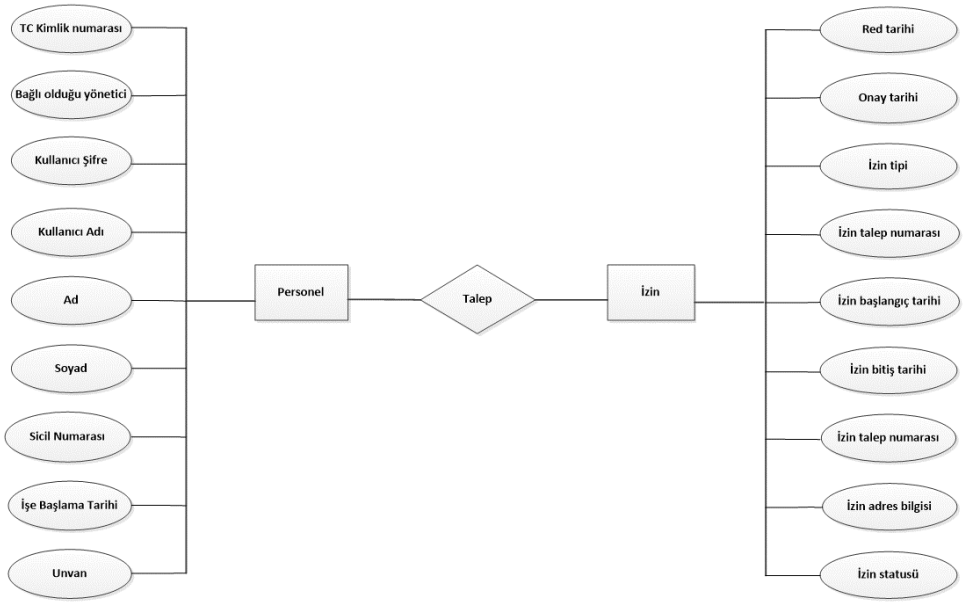


Şekil 5.2.2.2. İzin Talep Onay Ekranı

Şekil 5.2.2.3. de "İzinli Personel Listesi Ekranı" görülmektedi.

Şekil 5.2.2.3. İzinli Personel Listesi Ekranı

Arayüzler, arayüzlerdeki alanlar ve iş akışları analiz edildikçe yavaş yavaş sistemin veri modeli ve veri yapısı ortaya çıkmaya başlamıştır. Bu aşamada yapılan kademeli önce veri tablolarına esas olacak varlıklar ve ardından ilişkiler ile alanlar tespit edilmiştir.



Şekil 5.2.2.4. İzin Yönetim Sistemi Veri Yapısı

Bu aşamadan sonra detay analizin sistem analizi aşaması tamamlanmış olup ve modül analizi aşamasına geçilmiştir. Gerçekleştirilen çalışmada modül analizi aşamasında belirlenen İÖB'ler 24 adet olarak aşağıda listelenmiştir.

1. İzin talep formu doldur,
2. İzinli gün hakkı görüntüle,
3. İzin talep formu yazdır,
4. İzin talebi onaya gönder,
5. İzin talebi onaya gönder-yönetici izni,
6. Onay bekleyen izin talebi görüntüle,
7. İzin talebi yönetici onay/reddi ver,
8. İzin talebinin ik uzmanına yönlendir,

9. Eskalasyon loglarını görüntüle,
10. İzin talebine ık onayı/reddi ver,
11. Günlük izinli personel listesi görüntüle,
12. Günlük izinli personel listesi kaydet,
13. İzin talep geçmişini görüntüle,
14. İzin talebini iptal et,
15. İzin bakiyelerini otomatik olarak güncelle,
16. Personel izin bakiyesine manuel güncelle,
17. Sisteme giriş yap,
18. Sistemden güvenli çıkış,
19. Active directory rolleriyle sistem rollerini eşleştir,
20. Kişi bazında işlem logu görüntüle,
21. İşlem bazında log detayı görüntüle,
22. Durum aksiyon özetini görüntüle,
23. Durum değiştirme ve bildirim yollama,
24. Sistem ayarlarını güncelle.

Her bir İÖB için kullanacağı arayüz, iş kuralları, veri modeli, veri yapısı, İOG ve akış detaylı olarak tanımlanmıştır. Aşağıda örnek olarak personelin kendi izin geçmişini görüntülediği "İzin Geçmişini Görüntüle" İÖB bilgileri görülmektedir.

Çizelge 5.2.2.1. İzinli Geçmişini Görüntüle İÖB Detay Kartı

Başlık	**Açıklama**
İşlevsel Öz Birim	İzin geçmişini görüntüle
Açıklama	Personel kendine ait izin geçmişini listesini görüntüler
Ekran	İzin talep geçmişi ekranı
İş Kuralı	İzin talepleri en son izin talebinden geçmişe doğru sıralı gelmelidir.
İşlem Tipi	Veri okuma
Veri Modeli	Kullanıcı, İzin Talep
Veri Yapısı	İzin talep tarihi, izin onay/red tarihi, izin başlangıç tarihi, izin bitiş tarihi, izin süresi, kalan izin bakiyesi, açıklama, onaylayan yönetici
İşlevsel Olmayan Gereksinimler	(yok)
İş Akışı / Senaryo	İzin talep yönetimi
Kullanıcı Rolü	Personel
Alt sistem / modül	İzin Talep Modülü

5.2.3. Örnek Uygulamanın Karmaşıklık/Büyüklük ve Emek Tahminlemesi

Örnek uygulama için belirlenen 14 İÖB için karmaşıklık puanları verilerek toplam 165,5 puan hesaplanmıştır. Bu puan için belirlenen adam-saat çarpanı her bir puan için 1 saat olarak tahmin edilmiş ve sonuç olarak işin 165,5 saatte yani yaklaşık 21 günde bitirileceği ön görülmüştür. Örnek uygulama için çalışacak proje ekibinin eski yöntemleriyle mevcut analiz üzerinden tahminleme yapmaları istenmiş ve 20 günlük bir tahmin alınmıştır.

Proje ekibi analizi teslim ettikten 7 hafta sonra uygulamayı devreye almış ve 24 adam günlük bir emek gerçekleştirdiklerini ifade etmişlerdir. Bu sonuçla birlikte tahminleme sonucu aşağıdaki tabloda gösterildiği gibidir.

Çizelge 5.2.3.1. İzinli Personel Listesi Ekranı

Açıklama	**Değer**
Toplam Puan	166
Puan başına ön görülen emek (adam saat)	1
Proje için ön görülen toplam emek (adam saat)	166
Proje için ön görülen toplam emek (adam gün) (günlük 8 saat çalışma esas alınmıştır.)	21
Projenin gerçekleşme süresi (adam gün)	24
Tahminlenen ile gerçekleşen arasındaki sapma oranı	%20
Gerçekleşene göre puan başına emek (adam saat)	1,2

5.3. Personel Talep Yönetimi Uygulaması

Model iyileştirilip, modeli uygulamak üzere bir yazılım geliştirildikten sonra, iyileşmenin etkilerini ve geliştirilen yazılımın verimliliğini gözlemlemek üzere üçüncü bir proje üzerinde çalışma yapılmıştır. Örnek çalışmanın gerçekleştirildiği firmadaki personelin insan kaynakları ile iletişimini sağlayacak, talep ve şikayetlerini yönetmek üzere geliştirilecek olan bu yazılım projesi "Personel Talep Yönetimi" olarak isimlendirilmiştir.

Projede yeni analiz yaklaşımının kullanılmasıyla birlikte aynı zamanda bu yaklaşımın uygulanması kolaylaştırmak amacıyla geliştirilmiş olan "CFUWin" yazılımından da faydalanılmıştır.

5.3.1. “Personel Talep Yönetimi” Uygulamasının Arka Planı

Örnek uygulama için çalışılan kurumun İnsan Kaynakları departmanına gelen bordro dökümü, vize için gerekli belgeler, kredi için gerekli bilgiler, çalışan belgesi gibi çeşitli belge talepleri ve personel şikayetlerinin izlenebileceği bir otomasyon yazılımı gerçekleştirilmesi yönünde bir talebi söz konusu olmuştur. İK departmanının bu yazılımdan temel beklentisi, İK uzmanının kendine gelen talebi belirlenen SLA (servis seviyesi anlaşması) süresi içinde en efektif şekilde karşılaması şeklindedir.

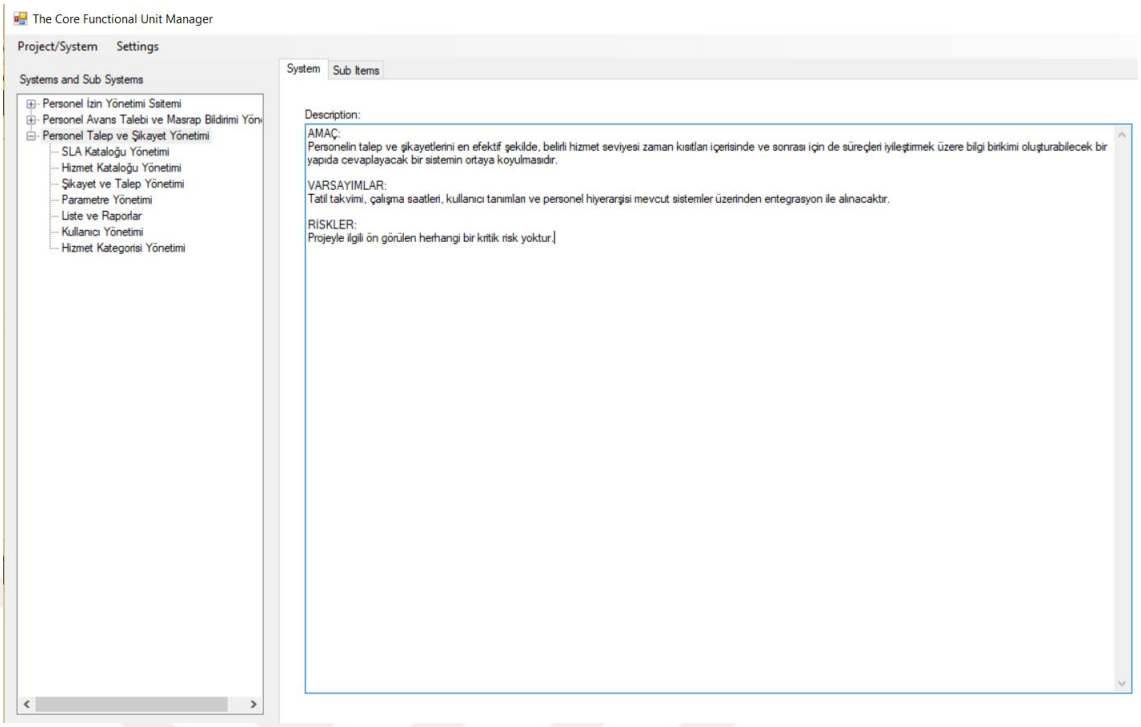
Uygulamanın çalışma temeli bir personelin kendisi ya da bir başkası için talep açmasına dayanmaktadır. Açılan bu talep yetki verilen İK çalışanlarının erişebildiği bir havuza düşer. Talebin açılmasıyla birlikte talebin kategorisine bağlı bir SLA cevap süresi için devreye girer. Tanımlı süre içinde cevap verilmemesi durumunda İK yöneticisine bildirim yollanır. Talep belirli bir İK uzmanı tarafından ele alınarak değerlendirilir ve gerekli hazırlıklar varsa yapılarak talebi açana bir cevap verilir. Böylece talep karşılanmış olacaktır. Talep bir İK çalışanı tarafından üzerine alındığında da başka bir SLA devreye girer.

Taleple ilgili talebi açan ve karşılayan arasında yazışmalar yapılabilir ve bu yazışmalar ve yapılan işlemler bir tarihçe olarak kayıt altına alınmalıdır.

Ayrıca uygulama bir web projesi olmalı ve hem intranet hem de internet üzerinden erişilebilmelidir.

5.3.2. “Personel Talep Yönetimi” Analizi

Uygulamanın analizi hem belirtimi hem analiz unsurları hem de tahminleme açısından güncellenen versiyonuyla İÖB modeli esas alarak yürütülmüştür.

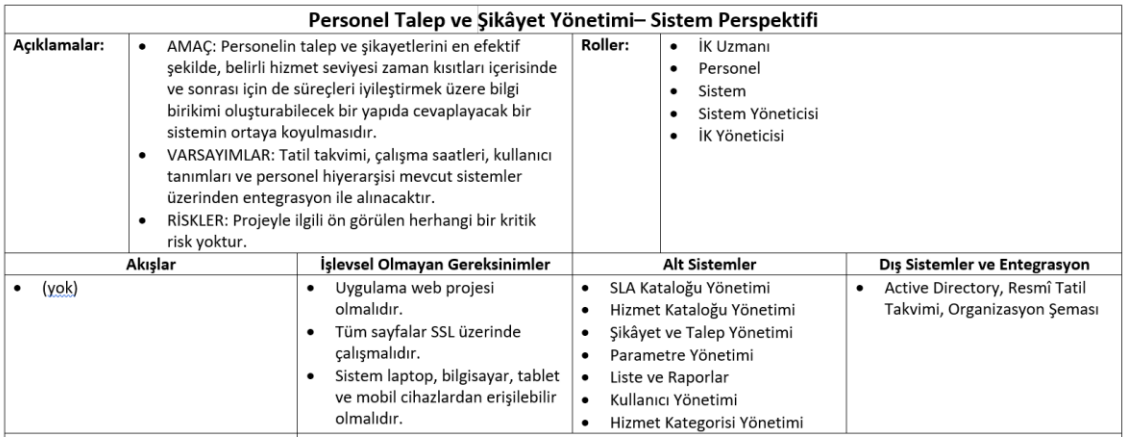


Şekil 5.3.2.1. Personel Talep ve Şikâyet Yönetimi Projesi Sistem Detayı Ekranı

Sistemin amaç ve varsayımları belirlenmiş, ancak Şekil 5.3.2.1. de görüleceği üzere yönetilmesi gereken kritik bir risk öngörülmemiştir. Sistem aşağıda listelenen 7 alt sistemden oluşmaktadır:

1. SLA Kataloğu Yönetimi,
2. Hizmet Kataloğu Yönetimi,
3. Şikâyet ve Talep Yönetimi,
4. Parametre Yönetimi,
5. Liste ve Raporlar,
6. Kullanıcı Yönetimi,
7. Hizmet Kategorisi Yönetimi.

Analizin ilk aşamalarında yukarıdaki bilgilerden başka roller, akışlar, İOG'ler ve dış sistem entegrasyonlarının analizi yapılarak Şekil 5.3.2.2. deki sistem perspektif dokümanı elde edilmiştir.



Personel Talep ve Şikâyet Yönetimi– Sistem Perspektifi			
Açıklamalar:	• AMAÇ: Personelin talep ve şikayetlerini en efektif şekilde, belirli hizmet seviyesi zaman kısıtları içerisinde ve sonrası için de süreçleri iyileştirmek üzere bilgi birikimi oluşturabilecek bir yapıda cevaplayacak bir sistemin ortaya koyulmasıdır. • VARSAYIMLAR: Tatil takvimi, çalışma saatleri, kullanıcı tanımları ve personel hiyerarşisi mevcut sistemler üzerinden entegrasyon ile alınacaktır. • RİSKLER: Projeyle ilgili ön görülen herhangi bir kritik risk yoktur.	**Roller:**	• İK Uzmanı • Personel • Sistem • Sistem Yöneticisi • İK Yöneticisi
Akışlar	**İşlevsel Olmayan Gereksinimler**	**Alt Sistemler**	**Dış Sistemler ve Entegrasyon**
• (yok)	• Uygulama web projesi olmalıdır. • Tüm sayfalar SSL üzerinde çalışmalıdır. • Sistem laptop, bilgisayar, tablet ve mobil cihazlardan erişilebilir olmalıdır.	• SLA Kataloğu Yönetimi • Hizmet Kataloğu Yönetimi • Şikâyet ve Talep Yönetimi • Parametre Yönetimi • Liste ve Raporlar • Kullanıcı Yönetimi • Hizmet Kategorisi Yönetimi	• Active Directory, Resmî Tatil Takvimi, Organizasyon Şeması

Şekil 5.3.2.2. Personel Talep ve Şikâyet Yönetimi Projesi Sistem Perspektif Dokümanı

Perspektif dokümanından da görüleceği üzere sistemde 5 adet rol, 3 adet İOG ve 3 adet dış sistem entegrasyonu yer almaktadır.

Analizin ilerleyen aşamalarında yapılan çalışmalarla sistemin toplam 44 adet İÖB'den oluşması gerektiği tespit edilmiştir. Belirlenen İÖB'ler için belirlenen ekran sayısı 25 ve bildirim sayısı 5 adettir. Ayrıca bu 44 İÖB 61 adet işlevsel gereksinimi yerine getirmekte ve bunun için de 9 adet ilişkisel tablo kullanmaktadır. Aşağıda bu öğelerden bazıları yer almaktadır.

İşlevsel Öz Birimlerden Örnekler

1. SLA Ekle,
2. SLA Güncelle,
3. SLA Sil,
4. SLA Listele,
5. SLA Görüntüle,
6. SLA Aktive Et,
7. SLA Pasifleştir,
8. Yeni Kullanıcı Ekle,
9. Mevcut Kullanıcı Bilgilerini Güncelle,

10. Mevcut Kullanıcıyı Sil,
11. Mevcut Kullanıcıları Görüntüle,
12. Kullanıcı Parolası Sıfırla.

Kullanıcı Etkileşimi Öğelerinden Örnekler

1. Kullanıcı ve Rol Yönetim Ekranı,
2. Kullanıcı Eklem/Güncelleme Ekranı,
3. Kullanıcıya Ait Rollerin Yönetimi Ekranı,
4. Role Dahil Kullanıcıların Yönetimi Ekranı,
5. Active Directory Kullanıcı Arama Ekranı,
6. Sisteme yeni bir kullanıcı olarak eklendiniz bildirimi,
7. Hesabınız aktifleştirildi bildirimi,
8. Hesabınız pasifleştirildi bildirimi,
9. Kullanıcı bilgileriniz güncellendi bildirimi,
10. Rol ve yetkileriniz güncellendi bildirimi,
11. Hesabınız silindi bildirimi,
12. Parolanız sıfırlandı bildirimi,
13. SLA listeleme ekranı,
14. SLA detay ekranı,
15. SLA güncellendi bildirimi,
16. SLA aktive edildi bildirimi.

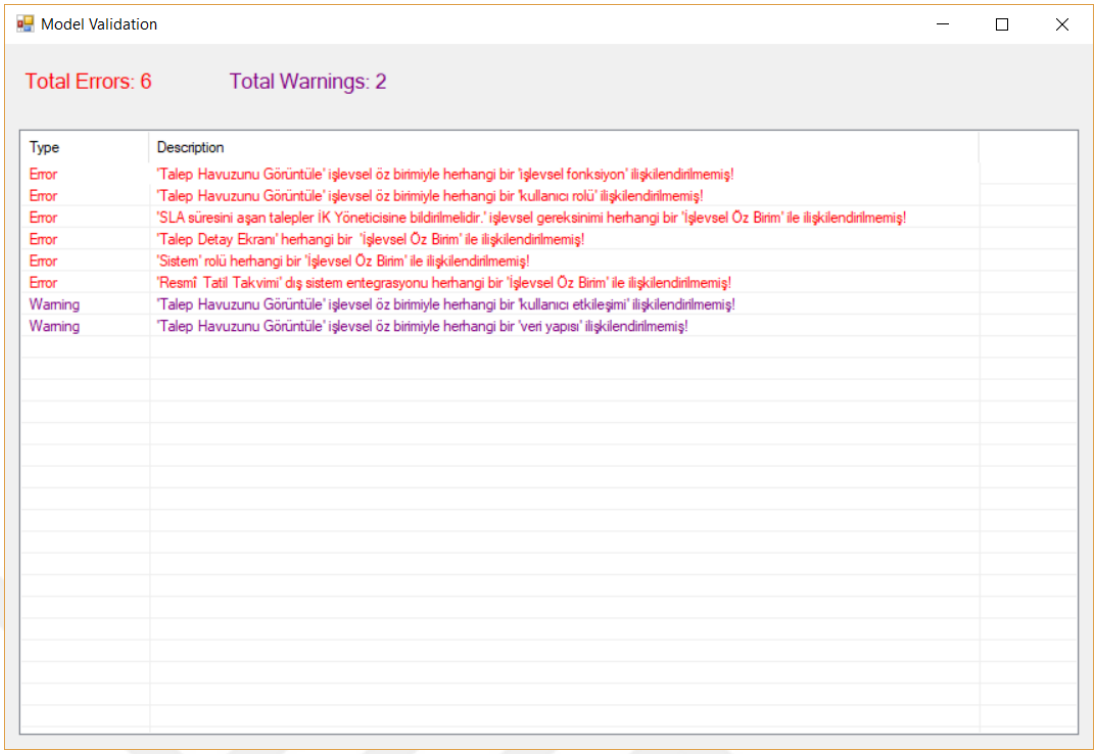
İşlevsel Gereksinimlerden Örnekler

1. Yetkili kullanıcı yeni bir SLA tanımlayabilmelidir,
2. Yetkili kullanıcı mevcut bir SLA'i güncelleyebilmelidir,
3. Yetkili kullanıcı mevcut bir SLA'i silebilmelidir,
4. Yetkili kullanıcı mevcut bir SLA'i görüntüleyebilmelidir,

5. Yetkili kullanıcı mevcut SLA'leri listeleyebilmelidir,
6. Yetkili kullanıcı aktif bir SLA'i pasifleştirebilmelidir,
7. Yetkili kullanıcı pasif bir SLA'i aktifleştirebilmelidir,
8. Sistemde herhangi bir konuda cevap için ayrı çözüm için ayrı SLA'ler tanımlanabilmelidir,
9. SLA takibi yapılırken resmî tatiller ve çalışma saatleri göz önünde bulundurulmalıdır,
10. Sisteme Active Directory de tanımlı biri kullanıcı yeni bir kullanıcı olarak eklenebilmelidir,
11. Mevcut kullanıcının bilgileri güncellenebilmelidir,
12. Mevcut bir kullanıcı sistemden silinebilmelidir. (Active Directory den silinmeyecek.)

5.3.3. Analiz Doğrulama ve Tahminleme

Analiz süreci boyunca tanımlanan öğelerle ilgili herhangi bir eksik olup olmadığını görmek amacıyla WinCFU yazılımının Şekil 5.3.3.1. de bir örneği sunulan model doğrulama çıktıları kullanılmıştır.



Şekil 5.3.3.1. CFUWin Model Doğrulama Ekran Görüntüsü

Model doğrulama listesinden edinilen bilgiler doğrultusunda eksikler giderilerek analiz hataları minimuma indirilerek çalışmalara devam edilmiştir.

Analizin sonunda ortaya koyulan 44 adet İÖB için WinCFU üzerinden karmaşıklık ve emek tahmini çıktıları alınmıştır.

	A	B	C	D	E	F	G	H	I	J	K	L
1	ID	SystemID	Title	İG	İG. K.	UI	UI. K.	Data	Data K.	İÖBKHP	İOGK	İÖBNKP
20	83	18	Talep İptal Et	1	1,20	1	1	1	0,80	3	1	3
21	84	18	Talep Tarihçesi Görüntüle	2	1,20	3	1	1	0,80	6,2	1	6,2
22	49	16	SLA Ekle	2	1,20	2	1	2	0,80	6	1	6
23	50	16	SLA Güncelle	2	1,20	2	1	2	0,80	6	1	6
24	52	16	SLA Listele	2	1,20	3	1	1	0,80	6,2	1	6,2
25	56	21	Yeni Kullanıcı Ekle	2	1,20	2	1	2	0,80	6	1	6
26	57	21	Mevcut Kullanıcı Bilgilerini Güncelle	3	1,20	2	1	2	0,80	7,2	1	7,2
27	63	21	Kullanıcı Rollerini Güncelle	3	1,20	2	1	2	0,80	7,2	1	7,2
28	64	21	Role Ait Kullanıcıları Güncelle	3	1,20	2	1	2	0,80	7,2	1	7,2
29	65	25	Hizmet Kategorisi Ekle	2	1,20	2	1	2	0,80	6	1	6
30	66	25	Hizmet Kategorisi Güncelle	3	1,20	2	1	2	0,80	7,2	1	7,2
31	69	25	Hizmet Kategorisi Görüntüle	2	1,20	3	1	1	0,80	6,2	1	6,2
32	72	17	Hizmet Ekle	2	1,20	3	1	2	0,80	7	1	7
33	73	17	Hizmet Güncelle	2	1,20	3	1	2	0,80	7	1	7
34	79	18	Talep Havuzunu Görüntüle	5	1,20	5	1	2	0,80	12,6	1	12,6
35	80	18	Talep Aç	7	1,20	4	1	5	0,80	16,4	1	16,4
36	81	18	Başkası İçin Talep Aç	3	1,20	3	1	2	0,80	8,2	1	8,2
37	82	18	Talep Güncelle	6	1,20	5	1	4	0,80	15,4	1	15,4
38	85	18	Talebe Not Ekle	2	1,20	2	1	2	0,80	6	1	6
39	86	18	Talep Detayı Görüntüle	2	1,20	3	1	1	0,80	6,2	1	6,2
40	87	19	Parametre Güncelle	2	1,20	2	1	2	0,80	6	1	6
41	88	18	SLA Süresi Aşanları Yöneticiye Bildir	5	1,20	2	1	1	0,80	8,8	1	8,8
42	89	20	Açık Talepleri Listele	2	1,20	3	1	1	0,80	6,2	1	6,2
43	90	20	Talep Detayını Raporla	5	1,20	3	1	2	0,80	10,6	1	10,6
44	91	20	Aylık Talep Özet Raporu	5	1,20	3	1	2	0,80	10,6	1	10,6
45												277,4

Şekil 5.3.3.2. Proje İÖB Bazında Karmaşıklık Hesaplama Cetveli

WinCFU tarafından üretilen MS Excel dokümanının bir bölümü Şekil 5.3.3.2. de görülmektedir. Tabloda İÖB'ler ile bunlara ait işlevsel gereksinim, kullanıcı etkileşimi, veri yapısı ve bunların faktör puanlarıyla çarpımları görülebilmektedir.

Hesaplanan 277,4 puanlık toplam İÖB karmaşıklığının üzerine öncelikle 50 puanlık entegrasyon puanının eklenildiği Şekil 5.3.3.3. de görülmektedir. Böylece tüm proje için toplam 327,4 puanlık bir karmaşıklık hesaplanmış olmaktadır.

Toplam İÖBNKP	277,4		İÖB Toplam karmaşıklığı
Entegrasyon Puanı	50		
PTKP	**327,4**		Toplam proje karmaşıklığı
KPAGDK	0,125		1 puan 1/8 adam-gün
BEBK	1		Başarım katsayısı
Topam Efor	**40,925**		adamgün

Şekil 5.3.3.3. Proje Emek Hesaplama Cetveli

Ardından bir puanın kurumda karşılığının kaç adam gün olduğu bilgisi ile proje ekibinin kurum ortalamasına göre başarım oranı kullanılarak gerçek emek tahmini ortaya koyulmuştur.

Uygulama tamamlandığında proje ekibinin ilerleme raporlarından projenin toplam gerçekleşen emeğinin 45 gün olduğu görülmüştür. Bu da sapmanın %8 civarında kaldığı anlamına gelir. Bu sapma daha önceki iki örneğe göre daha iyi bir sonuçtur. Ayrıca Molokken ve Jorgensen (2003) eski tarihli olmasına rağmen fikir vermesi açısından anlamlı olacak bir çalışmalarında yazılım projelerindeki ortalama sapmanın %30 olduğunu söylemektedirler.

5.4. Yapılan Çalışmanın Mevcut Modellerle Karşılaştırılması

Örnek uygulamalardan sonuncusu yapılırken, analiz aşamasında mevcut modellerden ikisiyle de analiz çalışmaları gerçekleştirilmiştir. Burada amaç önerilen yöntem ile diğer yöntemlerin bir gerçek yaşam uygulaması üzerinden kıyaslanabilmesidir.

Çalışmanın bu aşamasında mevcut modele alternatif olarak kullanıcı hikayesi ve geleneksel yöntemler ele alınmıştır. Çalışma boyunca elde edilen bilgiler ve analiz süreci önerilen model üzerinden uygulanırken belirtim tercih edilen bu yaklaşımlara göre gerçekleştirilmiştir. Analiz ve geliştirme tamamlandıkça proje ekibinin analiz yaklaşımlarıyla ilgili görüşleri bire-bir görüşme tekniğiyle alınmış ve yeni modelin mevcut modele göre güçlü ve zayıf yönleri tartışılmıştır.

5.4.1. Kullanıcı Hikayeleri ve Ürün Birikimi Listesi ile Analiz

Çevik yöntemlerde en çok uygulanan gereksinim analizi metotlarından biri kullanıcı hikayeleridir. Bir sistemin sahip olması gereken özellikler, yerine getirmesi gereken gereksinimler ve ihtiyaç duyulan diğer her şey kullanıcı hikayesi formatında yazılır ve ürün birikimi listesi adı verilen bir listeye dahil edilir.

Kullanıcı hikayeleri temelde 3 ana bölümden oluşur:

1. Kart (Card),
2. İletişim (Conversation),
3. Doğrulama/Kabul (Confirmation).

Kart, sistemden beklentisi olan bir rolün, sistemden ne beklediğinin ve bunu hangi amaçla beklediğinin yazılmasıdır.

"A rolü olarak sistemin B işlevini gerçekleştirmesini istiyorum ki, C ihtiyacım karşılansın."

Yukarıdaki cümle kullanıcı hikayesi için bir kart bilgisinin biçimidir. Gerçekleştirilen uygulama için verilebilecek bir kullanıcı hikayesi örneği şu şekilde olacaktır:

"Personel rolündeki bir kullanıcı olarak sistemde yeni bir şikâyet ya da talep açabilmeliyim ki, otomasyon üzerinden talebimin hızlıca yerine getirilmesi mümkün olsun."

Kullanıcı hikayesinin iletişim bölümünde analizi yapanın notları yer alır. Varsa kullanıcı hikayesiyle ilgili detay bilgiler, hesaplamalar, formüller, notlar buraya yazılır.

Kullanıcı hikayesinin doğrulama ve kabul bölümünde ise kabul kriterleri yer alır. Kullanıcı hikayesi gerçekleştirilip kodlandıktan sonra paydaşlar tarafından teslim alınmadan yapılacak testler, kontroller ve sınamalar burada yazılmalıdır.

Bu yöntemde uygulamayla ilgili bütün kullanıcı hikayeleri tespit edildikten sonra önceliklendirilerek sıralanmış ve böylece ürün birikimi listesi oluşturulmuştur. Aşağıda "Personel Talep ve Şikâyet Yönetimi" sistemi için oluşturulan ürün birikimi listesi yer almaktadır.

Personel Talep ve Şikâyet Yönetimi Kullanıcı Hikayeleri Listesi

1. Sistem Yöneticisi olarak sisteme yeni bir SLA ekleyebilmek istiyorum.
2. Sistem Yöneticisi olarak var olan bir SLA'i güncelleyebilmek istiyorum.
3. Sistem Yöneticisi olarak mevcut bir SLA'i silebilmek istiyorum.
4. Sistem Yöneticisi olarak mevcut SLA'leri listeleyebilmek istiyorum.
5. Sistem Yöneticisi olarak mevcut bir SLA'i listeden seçtikten sonra detaylı olarak görüntüleyebilmek istiyorum.
6. Sistem Yöneticisi olarak pasif olan bir SLA'i aktive edebilmek istiyorum.
7. Sistem Yöneticisi olarak aktif bir SLA'i pasife alabilmek istiyorum.

8. Sistem Yöneticisi olarak sisteme yeni bir kullanıcı ekleyebilmek istiyorum.
9. Sistem Yöneticisi olarak sistemde mevcut bir kullanıcının kullanıcı bilgilerini güncelleyebilmek istiyorum.
10. Sistem Yöneticisi olarak mevcut kullanıcıları listeleyebilmek istiyorum.
11. Sistem Yöneticisi olarak mevcut kullanıcıları görüntüleyebilmek istiyorum.
12. Sistem Yöneticisi olarak mevcut bir kullanıcıyı silebilmek istiyorum.
13. Bir kullanıcı olarak unuttuğum parolamı sıfırlayabilmek istiyorum.
14. Sistem Yöneticisi olarak pasif bir kullanıcıyı aktifleştirebilmek istiyorum.
15. Sistem Yöneticisi olarak aktif bir kullanıcıyı pasife alabilmek istiyorum.
16. Sistem Yöneticisi olarak mevcut kullanıcıların rollerini güncelleyebilmek istiyorum.
17. İK Uzmanı olarak mevcut hizmet kategorilerine yeni bir kategori ekleyebilmek istiyorum.
18. İK Uzmanı olarak hizmet kategorisi güncelleyebilmek istiyorum.
19. İK Uzmanı olarak mevcut bir hizmet kategorisini silebilmek istiyorum.
20. İK Uzmanı olarak mevcut hizmet kategorilerini listeleyebilmek istiyorum.
21. İK Uzmanı olarak mevcut bir hizmet kategorisi görüntüleyebilmek istiyorum.
22. İK Uzmanı olarak pasif bir hizmet kategorisi aktive etmek istiyorum.
23. İK Uzmanı olarak aktif bir hizmet kategorisi pasifleştirebilmek istiyorum.
24. İK Uzmanı olarak yeni bir hizmet ekleyebilmek istiyorum.
25. İK Uzmanı olarak mevcut bir hizmeti güncelleyebilmek istiyorum.
26. İK Uzmanı olarak mevcut bir hizmeti silebilmek istiyorum.
27. İK Uzmanı olarak mevcut hizmetleri listeleyebilmek istiyorum.
28. İK Uzmanı olarak mevcut bir hizmet detayını görüntüleyebilmek istiyorum.
29. İK Uzmanı olarak pasif bir hizmeti aktive edebilmek istiyorum.
30. İK Uzmanı olarak pasif bir hizmeti pasifleştirebilmek istiyorum.

31. İK uzmanı olarak talep havuzunu görüntüleyebilmek istiyorum.
32. İK Uzmanı olarak bana atanmış olan talepleri listeleyebilmek istiyorum.
33. İK Uzmanı olarak bana atanmış olan bir talebin detayını görüntülemek istiyorum.
34. İK Uzmanı olarak bana atamış bir talebi not ekleyebilmek istiyorum.
35. İK Uzmanı olarak bana atanmış bir talebin durumunu değiştirebilmek istiyorum.
36. Personel olarak sistemde yeni bir talep açabilmek istiyorum.
37. İK Uzmanı olarak sistemde başka bir personel için talep açabilmek istiyorum.
38. Personel olarak mevcut bir talebimi güncelleyebilmek istiyorum.
39. Personel olarak mevcut bir talebimi kapatabilmek istiyorum.
40. Personel olarak daha önceden gerçekleştirmiş olduğum taleplerin tarihçesini görüntüleyebilmek istiyorum.
41. Personel olarak mevcut aktif bir talebime not ekleyebilmek istiyorum.
42. Personel olarak mevcut bir talebimin detayını görüntüleyebilmek istiyorum.
43. Sistem Yöneticisi olarak mevcut parametreleri güncelleyebilmek istiyorum.
44. Sistem kullanıcısı olarak var olan SLA'leri İşletmek istiyorum.
45. İK Yöneticisi olarak aylık talepleri listeleyebilmek istiyorum
46. İK Yöneticisi olarak talep detayını raporlayabilmek istiyorum
47. İK Yöneticisi olarak aylık talep özet raporu görüntüleyebilmek istiyorum.
48. Personel olarak kullanıcı bilgilerimle sisteme giriş yapabilmek istiyorum.

Yukarıda listelen 48 öğeden oluşan ürün birikimi listesi kullanıcı hikayesinin en basit hali kullanılarak oluşturulmuştur. Çevik yaklaşımlar dokümantasyon yapılmamasına yönelik açıkça bir telkinde bulunmamakla birlikte, etkili bir dokümantasyonun nasıl yapılması gerektiğine dair de bir yöntem ortaya koymamaktadırlar. Çevik takımlarda müşteriyle yakın çalışmayla gerekli olan bilginin yüz yüze iletişimle alınması teşvik edilmektedir. Bu doğrultuda işlevsel gereksinimlerin

ürün birikimi listesiyle kapsandığı düşünülerek örnek uygulamayla kıyaslanmak üzere kullanıcı hikayelerinin bu hali üzerinden tartışılmasına karar verilmiştir.

5.4.2. Geleneksel Yöntemde İşlevsel Gereksinimler Listesi Üzerinden Analiz

Kullanıcı hikayesi ve kullanım vakası dışında yöntemlerle oluşturulan analiz dokümanlarının en temel öğesi işlevsel gereksinimler listeleridir. Bir yazılım sisteminin yerine getirmesi beklenen her türlü temel davranış, iş kuralı, sahip olması gereken özelliğin temel öğeleri birer işlevsel gereksinim olarak yazılır ve gerekiyorsa detaylandırılır. Bu tip dokümanlar genellikle SRS'i baz alır ve ona uygun olarak belirlenen başlıklarla işlevsel gereksinimler desteklenir. SRS başlıklarının hangi yöntemlerle ve hangi süreç adımlarıyla oluşturulacağına dair bir yönlendirme yoktur. Bununla birlikte kullanıcı etkileşimi, sistem ara birimi, iş akışları, işlevsel olmayan gereksinimler gibi birçok ek bilgiye yer verilmesi söz konusudur.

Mevcut modelin uygulama sonuçlarının sınanabilmesi için yapılan bu çalışmada "Personel Talep ve Şikâyet Yönetimi" sisteminin işlevsel gereksinimleri listelenmiştir.

1. Yetkili kullanıcı yeni bir SLA tanımlayabilmelidir.
2. Yetkili kullanıcı mevcut bir SLA'i güncelleyebilmelidir.
3. Yetkili kullanıcı mevcut bir SLA'i silebilmelidir.
4. Yetkili kullanıcı mevcut bir SLA'i görüntüleyebilmelidir.
5. Yetkili kullanıcı mevcut SLA'leri listeleyebilmelidir.
6. Yetkili kullanıcı aktif bir SLA'i pasifleştirebilmelidir.
7. Yetkili kullanıcı pasif bir SLA'i aktifleştirebilmelidir.
8. Sistemde herhangi bir konuda cevap için ayrı çözüm için ayrı SLA'ler tanımlanabilmelidir.
9. SLA takibi yapılırken resmî tatiller ve çalışma saatleri göz önünde bulundurulmalıdır.
10. Sisteme Active Directory de tanımlı biri kullanıcı yeni bir kullanıcı olarak eklenebilmelidir.

11. Mevcut kullanıcının bilgileri güncellenebilmelidir.
12. Mevcut bir kullanıcı sistemden silinebilmelidir. (Active Directory den silinmeyecek.)
13. Sistemdeki mevcut kullanıcılar listelenebilmelidir.
14. Sistemdeki bir kullanıcının parolası sıfırlanabilmelidir.
15. Hesabı pasif durumda olan bir kullanıcının hesabı aktifleştirilebilmelidir.
16. Sistemdeki bir kullanıcının hesabı pasif duruma alınabilmelidir.
17. Hesabı pasife alınan kullanıcı sisteme giriş yapamamalıdır.
18. Kullanıcının rolleri eklenip silinebilmelidir.
19. Bir role ait kullanıcılar eklenip silinebilmelidir.
20. Kullanıcı ekleme, silme, güncelleme, aktifleştirme, pasifleştirme, rol değişikliği ve parola sıfırlama işlemlerinde kullanıcıya bildirim gitmelidir.
21. Değişikliği yapılan bir SLA ile ilgili değişiklik, o SLA'yi aktif olarak kullanan öğeler için geçerli olmayacaktır. Yeni açılan bir talep ya da sorun için geçerli olacaktır.
22. SLA ancak daha önce hiç kullanılmadıysa silinebilir. Eğer kullanıldıysa pasifleştirilir.
23. Yetkili kullanıcı yeni bir Hizmet Kategorisi tanımlayabilmelidir.
24. Yetkili kullanıcı mevcut bir Hizmet Kategorisini güncelleyebilmelidir.
25. Yetkili kullanıcı mevcut bir Hizmet Kategorisini silebilmelidir.
26. Yetkili kullanıcı mevcut bir Hizmet Kategorisini görüntüleyebilmelidir.
27. Yetkili kullanıcı mevcut Hizmet Kategorilerini listeleyebilmelidir.
28. Yetkili kullanıcı aktif bir Hizmet Kategorisini pasifleştirebilmelidir.
29. Yetkili kullanıcı pasif bir Hizmet Kategorisini aktifleştirebilmelidir.
30. Yetkili kullanıcı yeni bir Hizmet tanımlayabilmelidir.
31. Yetkili kullanıcı mevcut bir Hizmeti güncelleyebilmelidir.

32. Yetkili kullanıcı mevcut bir Hizmeti silebilmelidir.
33. Yetkili kullanıcı mevcut bir Hizmeti görüntüleyebilmelidir.
34. Yetkili kullanıcı belirli bir kategoriye ait hizmetleri listeleyebilmelidir.
35. Yetkili kullanıcı aktif bir Hizmeti pasifleştirebilmelidir.
36. Yetkili kullanıcı pasif bir Hizmeti aktifleştirebilmelidir.
37. Bir hizmet en az bir kategoriye ait olmalıdır. Birden fazla kategoriye ait olabilir.
38. Hizmetin atanabileceği en az bir rol seçilmelidir. İstenirse birden fazla rol eklenebilmelidir.
39. Kullanıcılar sisteme düşen talepleri talep bir ekrandan izleyebilmelidir.
40. Talep durumları: gri-yeni talep, atanmamış, mavi-atanmış, SLA süresi içinde, turuncu-SLA süresinin %80'inde, kırmızı, SLA süresini geçmiş olarak görüntülenmelidir.
41. Sisteme yeni bir talep açıldığı an talep için SLA çalışmaya başlamalıdır.
42. Taleplere dosya eklenebilmelidir.
43. Bir kullanıcı başka bir personel için talep açabilmelidir. Bu durumda talebi açan ve talebin kimin için açıldığı bilgileri ayrı ayrı izlenebilmelidir.
44. Talep üzerinde talep sahibi ve İK uzmanları arasında mesajlaşma gerçekleştirilebilmelidir.
45. Sistem tarafından talep tarihçesi tutulmalıdır.
46. Kullanıcılar kendilerine ait talepleri geçmiş tarihliler de dahil olmak üzere listeleyebilmelidir.
47. Yetkili kullanıcılar sistem parametrelerini güncelleyebilmelidir.
48. SLA süresini aşan talepler İK Yöneticisine bildirilmelidir.
49. İK yöneticisi açık talepleri listeleyebilmelidir.
50. Yetkili kullanıcılar talep detayını yazdırabilmelidir.

51. Yetkili kullanıcılar Aylık Talep Özet Raporu görüntüleyebilmeli ve yazdırabilmelidir.
52. Kullanıcı kendi açtığı talepte yetkisi dahilinde güncelleme yapabilmelidir.
53. İK Uzmanı kendisine atanmış taleplerde yetkisi çerçevesinde güncelleme yapabilecektir.
54. Yapılan değişiklik SLA kurallarını etkileyecek yöndeyse, SLA sayacı buna göre güncellenmelidir.
55. Personel kendisi için yeni bir talep açabilmelidir.
56. Yetkili kullanıcı dilerse talebini geri çekip iptal edebilmelidir.
57. Talep iptalinde talebe cevap verme süresi de dahil SLA'ler hesaplamaya dahil edilmez.
58. Yetkili kullanıcı talebe not ekleyebilmelidir.
59. Yetkili kullanıcılar yetkileri dahilindeki detayları talep bilgilerini görüntüleyebilmelidir.
60. Yetkili kullanıcı geçerli bir kullanıcı adı ve parola ile sisteme giriş yapabilmelidir.
61. Talebe ekli bir dosya silinebilmelidir.

5.4.3. Yeni Model ile Kullanıcı Hikayesi ve Geleneksel Yöntem Yaklaşımlarının Karşılaştırılmaları

Yeni modelle gerçekleştirilen uygulamalar sonrasında, üçüncü uygulama için oluşturulan kullanıcı hikayeleri ve işlevsel gereksinim listesi de göz önünde bulundurularak projede çalışan analist, yazılım geliştirme ve test uzmanlarıyla genel bir değerlendirme çalışması gerçekleştirilmiştir. Bu değerlendirmede bire-bir görüşmelerle önerilen model, kullanıcı hikayeleri yöntemi ve geleneksel yaklaşımların güçlü ve zayıf yönleri sorulmuş ve alınan cevaplar çalışmanın tamamından elde edilen diğer bulgularla birlikte sonuçlar ve öneriler bölümünde sunulmuştur.

BÖLÜM 6

SONUÇLAR ve ÖNERİLER

Yazılım projeleri belirli bir iş ihtiyacını karşılamak, bir iş problemini çözmek ya da piyasadaki bir fırsatı yakalamak amacıyla gerçekleştirilirler. Her ne sebeple yapılıyor olursa olsun iyi bir yazılım üretmek ve başarılı bir proje geliştirmek tüm proje ekiplerinin birincil amacıdır. Yazılım proje başarıları üzerine yapılan çalışmalar maalesef bu alandaki başarı oranlarının beklenenin altında olduğunu göstermektedir. Yine benzer araştırmaların sonuçlarına göre başarı ve başarısızlık faktörlerinin en başında kullanıcı girdilerinin eksikleri, tamamlanmamış gereksinimler, gereksinim değişiklikleri, üst yönetim desteğinin yetersizliği bulunmaktadır. Araştırmalar göstermektedir ki daha başarılı projeler ve kaliteli yazılım ürünleri için gereksinim analizi alanına daha fazla önem vermek gerekmektedir.

Yazılım projelerindeki başarı ve başarısızlığı etkileyen en önemli unsurlardan biri olan gereksinim analizi yakından incelendiğinde, işin doğası gereği insanla yapılan çalışmalarda istek, ihtiyaç ve gereksinimlerin ediniminin zorluğunun yanı sıra, mevcut gereksinim analizi yöntemlerinin de yeterince olgunlaşmamış ve eksik olduğu gözlemlenmiştir. Mevcut yöntemlerde analizin nasıl bir süreçle yürütüleceğine, dokümanların nasıl olması gerektiğine, analizin nasıl doğrulanacağına, geliştirme emeğinin nasıl tahmin edileceğine dair ayrı ayrı çalışmalar olsa da bütüncül bir metodolojiye rastlanılmamıştır. Buna ek olarak tahminleme, analizin yürütülmesi ve yazılımın tasarım, geliştirme ve test öğeleriyle analiz unsurlarını eşleştirmek için atomik

bir analiz öğesi araştırıldığında da sadece kullanıcı hikayesi ve kullanım vakasıyla karşılaşılmıştır. Bunun haricinde en yaygın kullanılan analiz yöntemi olan metinsel analiz dokümanlarında da işlevsel gereksinimler, işlevsel olmayan gereksinimler ve diğer analiz bilgileri bir kompozisyon olarak sunulmakta ancak analiz için bir atomik öğe tanımlanmamaktadır. Araştırmalarda yazılım gereksinim analizinde kullanılabilecek onlarca farklı uygulamaya rastlanılmakla birlikte, uçtan uca bir analiz sürecini, sürecin adımlarını da destekleyecek şekilde bütünsel olarak geliştirilmiş bir araca rastlanmamıştır. Ayrıca incelenen araçların tamamı oluşturulan modelleri kendi içinde sınamakta, modellerin diğer analiz öğeleriyle ilişkisini doğrulamamaktadır. Örneğin akış diyagramı çizme araçları diyagramın kendi içerisinde doğrulaması yapmakta ancak akış öğelerinin işlevsel gereksinim, ekran ya da sistem kullanıcı rolleriyle ilişkisini kontrol etmemektedir.

Yapılan bu çalışmada özellikle gereksinim analizi sürecine yönelik yeni bir belirtim modeli ortaya koyulması amaçlanmıştır. Ancak çalışmanın ilerleyen aşamalarında, ortaya konulacak olan belirtim modeliyle analizin nasıl yapılacağıyla ilgili bir öneride bulunmanın da faydalı ve gerekli olduğu düşünülmüştür. Belirtim modelinin ilk örneği ortaya çıkmaya başlayıp örnek uygulamalar üzerinden denendikten sonra yapılan değerlendirmede yeni bir modelin uygun bir tahminleme sistemiyle desteklenmesinin de çalışmaya dahil edilmesine karar verilmiştir. Tüm bunlara ek olarak analisti doğru şekilde yönlendirecek, analiz hatalarını öğeler arasındaki ilişkileri kontrolünü yaparak minimuma indirecek, dokümantasyon ve tahminlemeyi otomatikleştirecek bir yazılımın da modele katkısının yüksek olacağı düşünülerek çalışmaya dahil edilmiştir.

Çalışma boyunca yapılan literatür taramaları, okumalar, modelle ilgili geliştirme ve uygulama çalışmaları, bunların değerlendirilmesi ve tartışılması sonucunda aşağıdaki sonuçlara ulaşılmıştır.

İyi ve etkili bir analiz modeli birim analizin anlamlı en küçük öğesi olarak kullanılacak atomik bir birime dayanıyor olmalıdır. Bu atomik birim hem analiz sürecinin başarılı bir şekilde yürütülmesine yardımcı olacak hem de analiz sonrasındaki tasarım, geliştirme ve test süreçlerindeki öğelerle eşleştirmeyi sağlayarak proje süresince etki analizi, değişiklik ve kapsam yönetimi, regresyon testi gibi işlemlere katkı sağlayacaktır

İyi bir analiz modelinin birimin işlevsel gereksinimlerle birlikte işlevsel olamayan gereksinimleri de ele alması kaçınılmazdır.

Gereksinimleri iş kuralları, akışlar, kullanıcı ve sistem etkileşim öğeleri, veri öğeleri, roller gibi daha somut perspektiflerle ele alıyor olması tüm paydaşlarca anlaşılabilir olması açısından önemlidir.

Efektif bir analiz yönteminde olması gereken diğer bir özellik de emek tahmin etme yaklaşımına sahip olmasıdır.

Analiz sürecinde olası hataları ortadan kaldıracak ya da azaltacak etkili bir araç da doğrulama yaklaşımıdır. Analiz öğelerinin her birini ayrı ayrı ve ilişkili olanları birbirleriyle karşılaştırarak bir eksik olup olmadığını kontrol etmek ve varsa eksikleri ortaya koyabilmek önemli bir artı değerdir. Mevcut analiz belirtim modelleri ve metodolojilerinde bütünsel bir doğrulama yaklaşımına rastlanmamıştır. Ancak sınırlı öğelerde kullanıcı hikayesi, kullanım vakası, ürün birikimi listesi gibi birbirinden bağımsız olarak bulunmaktadır.

Önerilen modelin bütün bunlar dışında sayılabilecek diğer bir güçlü yönü ise analiz sürecinin büyük bölümünün uçtan uca işletilebileceği bir otomasyon yazılımına sahip olmasıdır. Mevcut analiz yaklaşımları için piyasada bulunan çok sayıda yazılım bulunmakla birlikte, bu yazılımlar belirli bir tekniği gerçekleştirebilmekte ya da modellemeye yönelik olmaktadır.

Yukarıdaki belirtilen güçlü yönlerinin dışında ortaya koyulan model çalışmanın başında belirlenen 20 gereksinimin 17'sini çok büyük oranda karşılamaktadır. Geri kalan 3 gereksinim ikisini direkt olarak yerine getirmese de dosya ekleme mantığıyla kapsam içine alabilmektedir. 20 numaralı gereksinim ise modelin gerçek yaşam uygulamaları üzerinden yeterince sayıda projede ve farklı kurumlarda uygulanmasından sonra netleşeceğinden bu aşamada bu gereksinim kesin olarak karşılandığını söylemekten kaçınılmıştır.

Aşağıda yapılan çalışma sonucunda bu gereksinimlerin karşılanma durumları ve Bölüm 4.1.1. de ortaya koyulan gereksinimler parantez içinde numaraları verilerek belirtilmiştir.

Model, kullanıcı ve sistem gereksinimleri bir arada ve birbirleriyle ilişkilerini de ortaya koyarak tarif etmektedir (1).

Model hem işletimi hem de dokümantasyonu açısından hem yeni geliştirilecek projelerde hem de mevcut yazılımlarla ilgili değişiklik taleplerinde kullanılabiliyor olmalıdır (2).

Model, geliştirilecek olan yazılımın diğer yazılımlarla olan entegrasyonunu sistem seviyesinde emek tahminleme hesabına da dahil ederek yönetebilmektedir (3).

Model, yazılım projesinin teknik olan (yazılım mimarı, yazılım uzmanı, test uzmanı) ve olmayan (iş birimi alan uzmanları, anahtar kullanıcılar, sponsor) tüm paydaşları tarafından makul düzeyde okunabilir ve anlaşılabilir bir yapıdadır (4).

Model, işlevsel olmayan gereksinimler olarak da bilinen ve çoğunlukla yazılımın kalite özellikleri olarak nitelendirilen, güvenlik, kullanışlılık, performans gibi kısıtları da belirtebilmektedir (5).

Analiz modeli birbiriyle ilişkili olan çok sayıda detaylı gereksinimi anlamlı bir şekilde ve kullanıcıya değer katacak bileşenler olarak gruplandırmaktadır. Böylece yazılımın kullanıcıya üreteceği değeri ortaya koyacak soyut bir kavram üzerinden gereksinimler tarif edilebilecektir. Aynı zamanda özellik seviyeleri çerçevesinden soyutlama yapılarak sistem farklı seviyelerde tarif edilebilecektir (6,7).

Model, kullanıcı ve sistem arasındaki etkileşimi tarif etmeye uygun bir yapıda geliştirilmiştir. Böylece analist kullanıcının hangi aksiyonları aldığında sistemin hangi cevapları vereceği ve sistemde oluşan bazı durumların kullanıcıya nasıl aktarılacağı, hangi ekranların ya da ekran bileşenlerinin kullanılacağını, kullanıcı veri girişlerinin nasıl doğrulanacağı ve istenildiği durumda mesajların ve bildirimlerin nasıl kullanılacağını ve neler içereceğini model üzerinden tanımlayabilmektedir (8, 11,12, 13).

Model, kullanıcının sistem üzerinde gerçekleştirmek istediği iş ve işlemlerle ilgili olası tüm senaryoları ve senaryoların birbirleriyle ilişkisini de kapsayacak bir yapıya sahiptir. Bununla birlikte sistem, alt sistem seviyesinde iş akışlarının belirtimine de destek vermektedir (9,10).

Model veri yönetiminde kullanılacak varlıkları, bu varlıkların birbirleriyle ilişkilerini ve niteliklerinin yapısını, verinin nasıl saklanacağını ve verinin statik olarak hangi kısıtlarla, iş kurallarıyla çalışacağını ortaya koymaktadır (14).

Model, tarif edilen ihtiyaçlar ve belirlenen gereksinimler doğrultusunda ortaya koyulacak olan problemin büyüklüğünü ve karmaşıklığını hesaplayacak bir tahminleme mekanizmasına sahiptir (15).

Model görünümler ve aksiyonlar perspektifinde rol tabanlı yetkilendirmeyi desteklemektedir (16).

Model, analizin yüksek seviyeli perspektif ve detay seviyede analiz belirtiminin yapabilmekte ve analizdeki her bir öğenin nasıl, ne zaman yapılacağını ve birbirleriyle olan ilişkisini tarif etmektedir (19).

Aşağıdaki iki gereksinim model tarafından direkt olarak karşılanmamakla birlikte, sistem ve altındaki tüm öğeler için dosya ekleme özelliğinin bulunması nedeniyle dolaylı da olsa kapsama alınabilmektedir.

Model, sistem içerisinde gerekli olan yerlerde durum ve geçiş tarifini yapabilmelidir (17).

Model, fiziksel tasarıma girmeden, problemin tüm boyutlarıyla büyük resim çerçevesinden de değerlendirilebilmesi için kavramsal tasarım seviyesinde bir modelleme de sunmalıdır. Bu kavramsal model içerisinde çözümün temel bileşenleri olan akışlar, iş kuralları, ekranlar, entegrasyonlar, özellikler, işlevler, veri modeli tarif edilebilmelidir (18).

Aşağıdaki gereksinimin yerine getirilebilip getirilmediği ise ancak modelin birçok farklı kurum ve projede uygulanmasından sonra söylenebilecektir.

Model, eksiksiz ve herkesçe aynı şekilde uygulanabilir olmalı ancak gerektiğinde projeye ve kuruma uygun bir şekilde esnetilebilmelidir. Bunun için model aynı zamanda belirtim yapısıyla birlikte analiz sürecini de tarif etmelidir (20).

Diğer taraftan bu çalışmanın devamı niteliğinde yapılabilecek çalışmalar ile değerlendirilmesi gereken bazı hususlar da mevcuttur.

İlk olarak analiz modelinin daha karmaşık ve büyük projelerde denenmesi, etkinliğini ölçmek açısından önemlidir. Bu tip projelerden alınacak geri bildirimler doğrultusunda İÖB ile diğer öğelerin ilişkisi tekrar değerlendirilerek modelin olgunlaştırılması sağlanabilir. Benzer şekilde çevik projeler için de uygulanabilirliği denenerek daha esnek bir yapıda etkinliği gözlemlenebilecektir.

Modelle ilgili bir başka gelişim alanı olarak, yazılımın geliştirilmesinden sonra yapılacak yazılım testine esas olacak şekilde bir test senaryosu ya da kabul kriteri yapısı kurgulanabilir. Yine İÖB temelinde ilgili İÖB'lerin nasıl test edilmesi gerektiğine yönelik bir yapı, önerilen modeli daha da güçlendirilebilecektir.

Diğer taraftan yazılım kullanmanın avantajıyla İÖB ve diğer öğeler için farklı nitelikler tanımlanarak kayıt altına alınabilir ve bunlar analiz dokümanlarına aktarılabilir. Yazılım üzerinde İÖB ye bağlı işlevsel gereksinimler, işlevsel gereksinimlerin kullanıldığı İÖB'ler gibi çok sayıda liste ve rapor oluşturularak sistemin ve sistem öğelerinin çok farklı perspektiflerden görülebilmesi sağlanabilecektir.

Tüm bu çalışma sonuçlarına ve bulgulara dayanarak, mevcut modellerle karşılaştırmaların sonucuna bakıldığında önerilen modelin yazılım projelerinde uygulanmasının proje başarısı ve ürün kalitesine önemli oranda katkıda bulunacağı düşünülmektedir.

KAYNAKLAR

A Guide to the Business Analysis Body of Knowledge (BABOK guide): Version 2.0. (2009). Toronto: IIBA.

A guide to the Business Analysis Body of Knowledge: V3. (2015). Toronto: International Institute of Business Analysis.

A Survey on Implicit Requirements Management Practices in Small and Medium-Sized Enterprises. (2017). Tehnicki Vjesnik- Technical Gazette, 24(Supplement 1). doi:10.17559/tv-20150823114946

Abran, A. (2010). Software Metrıcs And Software Metrology. New Jersey:Wiley.

Adzic, G. (2011). Specıficatıon By Example. New York:Manning Publications.

Agile Extension to the BABOK® Guide, v2. (2017). Toronto, Ontario: International Institute of Business Analysis.

Al-Badareen, A. B., Selamat, M. H., Jabar, M. A., Din, J., & Turaev, S. (2011). Software Quality Models: A Comparative Study. Software Engineering and Computer Systems Communications in Computer and Information Science, 46-55. doi:10.1007/978-3-642-22170-5_4

Alexander, I. & Beus-Dukic, L. (2009). Discovering Requirements. New Jersey:Wiley.

Alexander, I.F. & Stevens, R. (2002). Writing Better Requirements. London:Pearson Education Ltd.

Al-Rawas1, A. & Easterbrook, S. (1996). Communıcatıon Problems In Requırements Engıneerıng: A Fıeld Study. First Westminster Conference on Professional Awareness in Software Engineering, Royal Society, London, 1-2 February.

Apel, S., & Kästner, C. (2009). An Overview of Feature-Oriented Software Development. The Journal of Object Technology, 8(5), 49. doi:10.5381/jot.2009.8.5.c5

Arora, A. & Bansal, S. (2005). Comprehensive Computer and Languages. New Delhi: Laxmi Publications.

Aurum, A. & Wohlin, C. (2005). Engineering and Managing Software Requirements. New York:Springer.

Beatty, J. & Anthony, C. (2012). Visual Models For Software Requirements, USA:Microsoft Press

Berson, A. & Dubov, L. (2007). Master Data Management and Customer Data Integration for a Global Enterprise, New York:The McGraw Hill Companies.

Bézivin, J., & Heckel, R. (2006). Guest Editorial to the Special Issue on Language Engineering for Model-Driven Software Development. Software & Systems Modeling, 5(3), 231-232. doi:10.1007/s10270-006-0028-6.

Bittner, K. & Spence, I. (2002). Use Case Modeling. New Jersey:Addison Wesley.

Boehm, B., Abts, C., & Chulani, S. (2000). Annals of Software Engineering, 10(1/4), 177-205. doi:10.1023/a:1018991717352

Bourque, P., & Fairley, R. E. (2014). SWEBOK: Guide to the software engineering body of knowledge. Los Alamitos, CA: IEEE Computer Society.

Boyer, J. & Mili, H. (2011). Agile Business Rule Development. New York:Springer.

Bozkus, Z., Bisson, C., & Arsan, T. (2009). Analytical expense management system. 2009 First International Conference on Networked Digital Technologies. doi:10.1109/ndt.2009.5272148

Bragança, A., Machado, R.J. (2006). Extending UML 2.0 Metamodel for Complementary Usages of the «extend» Relationship within Use Case Variability Specification. 10th International Software Product Line Conference (SPLC'06) 0-7695-2599-7/06

Cadle, J, Paul D., & Turner, P. (2010). Business Analysis Techniques, London:British Informatics Society Limited.

Castillo, I., Losavio, F., Matteo, A., & Bøegh, J. (2010). REquirements, Aspects and Software Quality: The REASQ model. The Journal of Object Technology, 9(4), 69. doi:10.5381/jot.2010.9.4.a4

Cockburn, A. (2000). Writing Effective Use Cases. New Jersey:Addison Wesley.

Çamoğlu, K., Akbayır, D. Yücalar, F. & Bayraklı, S. (2001). Bir Çevik Yazılım Geliştirme Sürecinin Uyarlanması Ve Uygulanması, Havacılık ve Uzay Teknolojileri Dergisi, Cilt 4, Sayı 3, Sayfa 57-67

Diev, S. (2006). Use cases modeling and software estimation. ACM SIGSOFT Software Engineering Notes, 31(6), 1. doi:10.1145/1218776.1218780

Dodani, M. H. (2008). The Architecture of Business. The Journal of Object Technology, 7(4), 43. doi:10.5381/jot.2008.7.4.c5

Elahi, G., Yu, E., Li, T., & Liu, L. (2011). Security Requirements Engineering in the Wild: A Survey of Common Practices. 2011 IEEE 35th Annual Computer Software and Applications Conference. doi:10.1109/compsac.2011.48

Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams, http://www.shengdongzhao.com/publication/elastic-hierarchies-combining-treemaps-and-node-link-diagrams/, son erişim 12 Mayıs 2019

Eriksson, M., Borg, K., & Börstler, J. (2006). 7.2.1 The FAR Approach - Functional Analysis/Allocation and Requirements Flowdown Using Use Case Realizations. INCOSE International Symposium, 16(1), 950-964. doi:10.1002/j.2334-5837.2006.tb02793.x

Firesmith, D. (2003). Modern Requirements Specification. The Journal of Object Technology, 2(2), 53. doi:10.5381/jot.2003.2.2.c6

Firesmith, D. (2003). Security Use Cases. The Journal of Object Technology, 2(3), 53. doi:10.5381/jot.2003.2.3.c6

Firesmith, D. (2003). Specifying Good Requirements. The Journal of Object Technology, 2(4), 77. doi:10.5381/jot.2003.2.4.c7

Firesmith, D. (2004). Creating a Project-Specific Requirements Engineering Process. The Journal of Object Technology, 3(5), 31. doi:10.5381/jot.2004.3.5.c4

Firesmith, D. (2004). Generating Complete, Unambiguous, and Verifiable Requirements from Stories, Scenarios, and Use Cases. The Journal of Object Technology, 3(10), 27. doi:10.5381/jot.2004.3.10.c3

Firesmith, D. (2004). Prioritizing Requirements. The Journal of Object Technology, 3(8), 35. doi:10.5381/jot.2004.3.8.c4

Firesmith, D. (2004). Specifying Reusable Security Requirements. The Journal of Object Technology, 3(1), 61. doi:10.5381/jot.2004.3.1.c6

Firesmith, D. (2005). Are Your Requirements Complete? The Journal of Object Technology, 4(1), 27. doi:10.5381/jot.2005.4.1.c3

Firesmith, D. (2005). Quality Requirements Checklist. The Journal of Object Technology, 4(9), 31. doi:10.5381/jot.2005.4.9.c4

Firesmith, D. (2006). Requirements Engineering Tasks. The Journal of Object Technology, 5(8), 21. doi:10.5381/jot.2006.5.8.c3

Firesmith, D. (2007). Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them. The Journal of Object Technology, 6(1), 17. doi:10.5381/jot.2007.6.1.c2

Gašević, D., Kaviani, N., & Hatala, M. (n.d.). On Metamodeling in Megamodels. Model Driven Engineering Languages and Systems Lecture Notes in Computer Science, 91-105. doi:10.1007/978-3-540-75209-7_7

Georgsson, A.(2011). Introducing Story Points and User Stories to Performe Estimations in a Software Development Organisation. (Yayımlanmamış tez çalışması.) Umeå University/ İsveç.

Griffiths, M. (2012). PMI-ACP Exam Prep. Minneapolis:RMC Publications.

Haggett P. & Chorkey, R.J. (1967). Models, Pradigs and the New Geography, Socioeconomic Models in Gagraphy.

Halle, B. (2002). Business Rules Applied -Business Better Systems Using the Business Rules Approach, New Jersey:Wiley.

Hammarberg, M. (2014). Joakım Sundén, Kanban in Action. New York:Manning Publications.

Harel, D., & Rumpe, B. (2004). Meaningful modeling: Whats the semantics of "semantics"? Computer, 37(10), 64-72. doi:10.1109/mc.2004.172

http://www.projectsmart.co.uk/docs/chaos-report.pdf, The Standish Group Chaos Report 2014, son erişim 14 Mayıs 2019.

http://www.versionone.com/assets/img/files/ChaosManifest_2011.pdf, The Standish Group Chaos Manifesto 2011, son erişim 1 Mayıs 2019

http://www.versionone.com/assets/img/files/ChaosManifesto2013.pdf, The Standish Group Chaos Manifesto 2013, son erişim 10 Mayıs 2019

Hull, E., Jackson, K., & Dick, J. (2005). Requirements Engineering Second Edition, New York:Springer.

IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE-STD 610.12-1990.

IEEE (1998). IEEE Guide for Developing System Requirements Specifications. IEEE-STD 1233-1998.

IEEE (1998). IEEE Guide for Software Requirements Specifications. IEEE Std 830-1998.

IEEE Software & Systems Engineering Standards Essential Collection https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/tocs/softess.pdf, son erişim 7 Mayıs 2019.

Johnson, R., Roussos, G., & Tagliati, L. V. (2008). Requirements analysis for large scale systems. The Journal of Object Technology, 7(8), 117. doi:10.5381/jot.2008.7.8.a3.

Jyoti Mahajan, Simmi Dutta, COREAN: A proposed Model for Predicting Effort Estimation having Reuse, International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-6, January 2013.

K. Beck, A. Cockburn, R. Jeffries, and J. Highsmith, Agile Manifesto, http://www.agilemanifesto.org, son erişim 20 Mayıs 2019.

Kan, S.H. (2002). Metrics and Models in Software Quality Engineering, Second Edition. New Jersey:Addison Wesley.

Kemerer, C. F. (1987). An empirical validation of software cost estimation models. Communications of the ACM, 30(5), 416-429. doi:10.1145/22899.22906.

Kreß Carl Friedrich, Hummel Oliver, Huq Mahmudul, A Practical Approach for Reliable Pre-Project Effort Estimation, CEUR Workshop Proceedings, Vol. 1138, p. 23, 2014.

Krishnamurthy, N. & Saran, A. (2008). Building Software A Practitioner's Guide. New York:Auerbach Publications.

Kruchten, P. (2003). Rational Unified Process, The: An Introduction, Third Edition. New Jersey:Addison Wesley.

Kulak, D. & Guiney, E. (2003). Use Cases: Requirements in Context, Second Edition. New Jersey:Addison Wesley.

Kühne, T. (2006). Clarifying matters of (meta-) modeling: An author's reply. Software & Systems Modeling, 5(4), 395-401. doi:10.1007/s10270-006-0034-8.

Larman, L. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition. New Jersey:Addison Wesley.

Lauesen, S. (2002). Software Requirements Styles and Techniques, New Jersey:Addison Wesley.

Leffingwell, D. & Widrig, D. (1999). Managing Software Requirements: A Unified Approach, New Jersey:Addison Wesley.

Leffingwell, D. & Widrig, D. (2003). Managing Software Requirements: A Use Case Approach, Second Edition. New Jersey:Addison Wesley.

Lilly, S. (n.d.). Use case pitfalls: Top 10 problems from real projects using use cases. Proceedings of Technology of Object-Oriented Languages and Systems - TOOLS 30 (Cat. No.PR00278). doi:10.1109/tools.1999.787547.

Loshin, D. (2009). Master Data Management. Massachusetts:Morgan Kaufmann Publishers.

Lucia, A. & Qusef, A. (2010). Requirements Engineering in Agile Software Development. Journal Of Emergıng Technologıes In Web Intellıgence. Vol. 2, No. 3, Ağustos.

Metz, P., Obrien, J., & Weber, W. (2003). Specifying Use Case Interaction: Types of Alternative Courses. The Journal of Object Technology, 2(2), 111. doi:10.5381/jot.2003.2.2.a1.

MIL-STD-498, Military Standard Software Devleopment and Documentation, USA Department of Defense, 1994.

Microsoft Corporation, MCSD Self-Paced Training Kit: Analyzing Requirements and Defining Microsoft .NET Solution Architectures, Exam 70-300, USA:Microsoft Press.2003.

Mike, C. (2004). User Stories Applied: For Agile Software Development. New Jersey:Addison Wesley.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63(2), 81-97. doi:10.1037/h0043158.

Molokken, K., & Jorgensen, M. (n.d.). A review of software surveys on software effort estimation. 2003 International Symposium on Empirical Software Engineering, 2003. ISESE 2003. Proceedings. doi:10.1109/isese.2003.1237981.

Norton, D. (2008). Agile Estimation and Planning Is Moving From a Dictatorship to a Democracy. Gartner.

Norton, D. (2009). Benchmarking Agile Productivity: Don't Assume Success, Prove It. Gartner.

Nuseibeh, B., & Easterbrook, S. (2000). Requirements engineering. Proceedings of the Conference on The Future of Software Engineering - ICSE 00. doi:10.1145/336512.336523.

Object Management Group, OMG Unified Modeling Language (OMG UML), Superstructure, v2.1.2, Tech rep., 2007.

Patton, J. (2014). User Story Mapping. California: O'Reilly Media.

Pei, Z., & Shao, Y. (2007). Extracting Linguistic Rules from Database using Linguistic Aggregation Operator. Proceedings on Intelligent Systems and Knowledge Engineering (ISKE2007). doi:10.2991/iske.2007.276.

Pham, A. & Pham, P. (2012). Scrum in Action: Agile Software Project Management and Development. Course Technology.

Pichler, R. (2010). Agile Product Management With Scrum. New Jersey:Addison Wesley.

Pohl, K., & Rupp, C. (2015). Requirements Engineering Fundamentals 2nd Edition, Rock Nook Inc.

Product Canvas, http://www.romanpichler.com/blog/the-product-canvas/, son erişim 3 Mayıs 2019.

Renown Microsoft Axapta Modules, http://www.renown.com.au/microsoft-dynamics/dynamics_ax/microsoft_dynamics_ax_modules.aspx, son erişim 11 Mayıs 2019.

Rinzler, B. (2009). Telling Stories, New Jersey:Wiley.

Robertson, S. & Robertson, J. (2006). Mastering the Requirements Process Second Edition. New Jersey: Addison Wesley Professional.

Robinson, W. N., & Elofson, G. (2004). Goal Directed Analysis with Use Cases. The Journal of Object Technology, 3(5), 125. doi:10.5381/jot.2004.3.5.a3.

Romi Satria Wahono, Analyzıng Requırements Engıneerıng Problems, IECI Japan Workshop, ISSN 1344-7491, pp. 55-58, 2003.

Rubin, K.S. (2013). Essential Scrum A Practical Guide To The Most Popular Agile Process. New Jersey:Addison Wesley.

S. Bose, M. Kurhekar, J. Ghoshal, Agile Methodology in Requirements Engineering, SETLabs Briefings Online, http://www.infosys.com/research/publications/agilerequirements- engineering.pdf, son erişim 20 Mayıs 2019.

Santillo, L. (2012). Easy Function Points -- Smart Approximation Technique for the IFPUG and COSMIC Methods. 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement. doi:10.1109/iwsm-mensura.2012.29.

Schleicher, A. & Westfechtel, B. (2001). Beyond Stereotyping: Metamodeling Approaches for the UML. 34th Hawaii International Conference on System Sciences.

Schmidt, M. (2000). Implementing the IEEE Software Engineering Standards. Indiana: Sams.

Schneider, F., & Berenbach, B. (2013). A Literature Survey on International Standards for Systems Requirements Engineering. Procedia Computer Science, 16, 796-805. doi:10.1016/j.procs.2013.01.083.

Selic, B. (2003). The pragmatics of model-driven development. IEEE Software, 20(5), 19-25. doi:10.1109/ms.2003.1231146.

Snyder,M., Steger, J., O'Brien, K., Landers, B. (2011). Microsoft® Dynamics™ CRM 4.0 and Microsoft Dynamics Live CRM Step by Step. USA:Microsoft Press.

Sommerville, I. & Pete, S. (1997). Requirements Engineering: A Good Practice Guide New Jersey:Wiley.

Specification Languages, http://en.wikipedia.org/wiki/-Category:Specification_languages, son erişim 7 Mayıs 2019.

The Standish Group, (2015), Chaos Report 2015, [PDF Dosyası], https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf, son erişim 5 Mayıs 2019.

Tse, T. H. (1991). An Examination of Requirements Specification Languages. The Computer Journal, 34(2), 143-152. doi:10.1093/comjnl/34.2.143.

Versionone, http://scrumgroup.org/wp-content/uploads/2015/03/state-of-agile-development-survey-ninth.pdf, 9th State of Agile Survey, , son erişim 2 Mayıs 2019.

Wallace, D., Raggett, I., & Aufgang, J. (2002). Extreme Programming for Web Projects. New Jersey:Addison Wesley.

Wiegers, K.E. (2003). Software Requirements, Second Edition. USA:Microsoft Press.

Wiegers, K.E. (2006). More About Software Requirements. USA:Microsoft Press.

Wikipedia, Front and Back Ends, http://en.wikipedia.org/wiki/- Front_and_back_ends, , son erişim 11 Mayıs 2019.

Windle, D.R. & Abreo, L.R. (202). Software Requirements Using the Unified Process: A Practical Approach. New Jersey: Prentice Hall PTR.

Withall, S. (2007). Software Requirement Patterns. USA:Microsoft Press.

Wright, J.M. (2011). A Modelling Language for Rich Internet Applications, (Yayınlanmamış doktora tezi.). Massey University/New Zeland.

Young, R.R. (2004). The Requirements Engineering Handbook. Massachusetts:Artech House.

ÖZGEÇMİŞ

1974 yılında İstanbul'da doğmuştur. 1996 yılı Anadolu Üniversitesi İktisat Fakültesi İktisat Bölümü mezunudur. 1995 yılından itibaren çeşitli kurumlarda yazılım projelerinde programcı, analist, proje yöneticisi ve birim yöneticiliği görevleri yürüttü. 2010 yılında Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans Eğitimini tamamladı. Çalıştığı dönem içinde ayrıca teknik eğitmenlik ve danışmanlık yapan Çamoğlu'nun veri tabanı, yazılım ve yazılım kalitesi hakkında beşi elektronik olmak üzere yayınlanmış 15 adet kitabı bulunmaktadır. Halen Trakya Üniversitesi'nde, Bilgisayar Mühendisliği alanında doktora eğitimini sürdürmekte olup, özel bir kuruluşta Ar-Ge ve İnovasyon Direktörü olarak görev yapmaktadır.