JOURNAL OF BUSINESS IN THE DIGITAL AGE

Vol. 8 Issue 2 2025

eISSN: 2651-4737 *dergipark.gov.tr/jobda*

Review Article

SİSTEM GELİŞTİRME YAŞAM DÖNGÜSÜ: TEORİDEN PRATİĞE SİSTEMATİK PROBLEM ÇÖZME YAKLAŞIMI

SYSTEM DEVELOPMENT LIFE CYCLE: A SYSTEMATIC PROBLEM-SOLVING APPROACH FROM THEORY TO PRACTICE

Vahap TECİM [1]

[1] Dokuz Eylül Üniversitesi, Yönetim Bilişim Sistemleri Bölümü, vahap.tecim@deu.edu.tr, ORCID: 0000-0001-5319-5241

Article Info:
Received: December 15, 2023
Revised 1: December 15, 2025
Revised 2: December 29, 2025
Accepted: December, 31, 2025

Anahtar Kelimeler:
Yönetim Bilişim Sistemleri,
Sistem Geliştirme Yaşam Döngüsü,
Sistem Yaklaşımı,
İş Zekası ve Analitiği,
Sistem Analizi ve Tasarımı

Keywords:
Management Information Systems,
Systems Development Life Cycle,
Systems Approach,
Business Intelligence and Analytics,
Systems Analysis and Design

DOI: 10.46238/jobda.1405138

ÖZ

Hızla değişen pazar dinamikleri ve karmaşıklaşan iş süreçleri, yöneticilerin klasik yöntemlerle karar almasını ve sorunları çözmesini yetersiz kılmaktadır. İşletmeler günümüzde anlık operasyonel hatalardan ziyade; süreç optimizasyonu, veri yönetimi ve stratejik belirsizlik gibi çok katmanlı kurumsal problemlerle mücadele etmektedir. Bu çalışma, söz konusu karmaşık problemlerin çözümünde bütüncül bir bakış açısı sunan Sistem Yaklaşımı ve Sistem Geliştirme Yaşam Döngüsü (SGYD) metodolojisini entegre eden bir yol haritası sunmaktadır.

Çalışma kapsamında öncelikle Genel Sistem Teorisi'nin kavramsal çerçevesi çizilmiş; donanım, yazılım, veri, insan ve süreç bileşenlerini, "Amaç ve Hedefler" doğrultusunda birleştiren modern Bilişim Sistemleri yapısı irdelenmiştir. Ayrıca geleneksel Yönetim Bilişim Sistemleri piramidi, İş Zekası ve İş Analitiği yetkinlikleriyle genişletilerek güncel bir perspektifle yeniden modellenmiştir. Çalışmanın uygulama boyutunda ise SGYD'nin aşamaları (Planlama, Analiz, Tasarım, Gerçekleştirme ve Bakım); soyut işletme problemlerinin somut ve sürdürülebilir bilişim çözümlerine dönüştürülmesinde bir kılavuz olarak ele alınmıştır.

SGYD'nin sadece bir yazılım üretim süreci değil, aynı zamanda stratejik bir problem çözme metodolojisi olduğu vurgulanan bu çalışmada; projelerin başarısızlık nedenleri, veri kalitesi ve kapsam yönetimi gibi kritik faktörler analiz edilmiştir. Sonuç olarak, bu sistematik yaklaşımın işletmelerin operasyonel verimliliğini artırmada, hata riskini minimize etmede ve rekabet avantajı sağlamada nasıl etkin bir araç olarak kullanılabileceği ortaya konulmuştur.

ABSTRACT

Rapidly changing market dynamics and increasingly complex business processes render classical decision-making and problem-solving methods insufficient for managers. Today, businesses grapple with multi-layered corporate problems such as process optimization, data management, and strategic uncertainty, rather than merely instantaneous operational errors. This study presents a roadmap integrating the Systems Approach and the Systems Development Life Cycle (SDLC) methodology, offering a holistic perspective for solving these complex problems.

First, the conceptual framework of General Systems Theory is outlined; subsequently, the modern Information Systems structure, which unites hardware, software, data, people, and process components under the guidance of "Goals and Objectives," is examined. Furthermore, the traditional Management Information Systems pyramid has been remodeled from a contemporary perspective by extending it with Business Intelligence and Business Analytics capabilities. Regarding the application dimension, the phases of the SDLC (Planning, Analysis, Design, Implementation, and Maintenance) are addressed as a guide for transforming abstract business problems into concrete and sustainable IT solutions.

Emphasizing that the SDLC is not merely a software production process but also a strategic problem-solving methodology, this study analyzes critical factors such as reasons for project failure, data quality, and scope management. In conclusion, the study demonstrates how this systematic approach can be utilized as an effective tool for increasing operational efficiency, minimizing error risks, and ensuring competitive advantage in businesses.

1 | GİRİŞ

Gerek kurumsal yapılarda stratejik kararlar alan yöneticiler, gerekse bilimsel araştırma yürüten akademisyenler; problemin tanımlanması, veri toplama yöntemleri, analiz süreçleri ve raporlama teknikleri gibi temel metodolojik sorunsallarla sıklıkla karşılaşmaktadır. Karar verme süreci, yalnızca yönetsel kademelerle sınırlı kalmayıp, sosyal yaşamın her alanında bireylerin maruz kaldığı karmaşık bir bilişsel süreçtir (Simon, 1977). Bireylerin en uygun alternatifi seçerek optimum faydayı sağlama çabası, günlük yaşamın rutin bir parçası haline gelmiştir.

Karar verme süreçlerinin her zaman rasyonel matematiksel modellere dayanmadığı, davranışsal iktisat teorileriyle de desteklenmektedir. Örneğin; bir üniversite öğrencisinin yemekhane tercihlerinde ekonomik avantaj (düşük maliyet) sunan haftalık abonelik yerine, daha maliyetli olan günlük ödemeyi tercih etmesi (bu oranın %30 seviyelerinde olması), karar süreçlerinde "sınırlı rasyonellik" ilkesinin ve psikolojik faktörlerin etkili olduğunu göstermektedir (Kahneman, 2011).

Bir problemin etkin çözümü; kök nedenlerin anlaşılmasını, ilişkili değişkenlerin tespit edilmesini ve çevresel koşullara (konjonktüre) uygun çözüm alternatiflerinin geliştirilmesini gerektirir. Probleme etki eden değişken sayısı arttıkça, çözümün karmaşıklığı da artmaktadır (von Bertalanffy, 1968). Bazı problemler lineer bir düzlemde çözülüp sonlanırken; bazıları döngüsel bir yapı arz eder. Elde edilen çıktı, bir sonraki sürecin girdisi haline gelerek sarmal bir geri bildirim (feedback) mekanizması oluşturur.

Sosyal bilimlerde ve iş hayatında problemlerin çözümü, yalnızca kantitatif (sayısal) verilerle ifade edilememektedir. Duygusal zeka, sezgiler ve sosyo-kültürel dinamikler, "doğru karar" kavramını "optimum karar" kavramına dönüştürmektedir. Özellikle kolektivist kültür özelliklerine sahip toplumlarda (Hofstede, 2001), yöneticilerin profesyonel kararlar alırken duygusal ve ilişkisel faktörlerden etkilenebildiği gözlemlenmektedir.

Literatürde problem çözme teknikleri ve sistem yaklaşımı üzerine çok sayıda çalışma bulunmakla birlikte; geleneksel Yönetim Bilişim Sistemleri (YBS) hiyerarşisinin günümüz veri analitiği yetkinlikleriyle (İş Zekası ve İş Analitiği) nasıl entegre edileceği ve bu yapının Sistem Geliştirme Yaşam Döngüsü (SGYD) içinde nasıl operasyonel hale getirileceği konusunda bir boşluk bulunmaktadır. Bu bağlamda çalışmanın temel araştırma problemi; karmaşıklaşan işletme problemlerinin çözümünde, klasik sistem teorisi ile modern analitik araçların bütünleşik bir modelde nasıl kullanılabileceğidir.

Bu doğrultuda çalışmanın temel amacı; her kademedeki çalışanın karşılaşabileceği problemlere uygulanabilecek kavramsal bir problem çözme modeli sunmaktır. Çalışma, türü itibariyle teorik çerçeve oluşturmaya yönelik kavramsal bir model önerisidir. Yöntem olarak; Genel Sistem Teorisi ve SGYD literatürü taranarak elde edilen veriler, betimsel analiz yöntemiyle yorumlanmış ve "Analitik Yetkinlik Tabanlı Genişletilmiş YBS Modeli" kurgulanmıştır.

Çalışmanın literatüre sağladığı özgün katkı; geleneksel Anthony Piramidi (YBS Piramidi) olarak bilinen yapının, güncel "İş Zekası" ve "İş Analitiği" kavramlarıyla yeniden modellenerek genişletilmesi ve bu yapının SGYD süreçleriyle ilişkilendirilmesidir. Bu yaklaşım, sadece teknik bir yazılım süreci olarak görülen SGYD'yi, stratejik bir yönetim aracı olarak yeniden konumlandırmaktadır.

Endüstri 4.0 ve dijital dönüşüm süreçleriyle birlikte insan hatasının minimize edildiği, otonom sistemlerin öne çıktığı günümüzde; kurumsal problemlere bilişim tabanlı ve sistematik çözümler geliştirilmesi kritik bir gereklilik halini almıştır (Schwab, 2016).

Ancak, bir metodolojiye körü körüne bağlı kalmanın da riskleri mevcuttur. Araştırmacılar veya uygulayıcılar, bazen asıl problemi çözmek yerine, seçilen metodolojinin adımlarını tamamlama tuzağına düşebilmektedir. Kaplan'ın (1964) da belirttiği gibi, "Elinizde sadece bir çekiç varsa, her sorun size bir çivi gibi görünmeye başlar". Dolayısıyla, en iyi metodoloji bile yanlış probleme uygulandığında başarısız olmaya mahkumdur. Bu çalışma, yöntemsel fanatizmden kaçınarak, problemin doğasına uygun esnek ve sistematik bir yaklaşımı savunmaktadır.

2 | SİSTEM YAKLAŞIMI

Bir problemi bütünüyle kavramak ve etkin bir çözüm üretmek, o problemi oluşturan unsurların ve aralarındaki ilişkilerin anlaşılmasını gerektirir. "Sistem" olarak ifade edilen bu ilişkiler bütününü göz ardı ederek kalıcı bir çözüm üretmek mümkün değildir. Genel Sistem Teorisi'nin öncüsü Ludwig von Bertalanffy'ye (1968) göre sistem; belirli bir amaca ulaşmak için bir araya gelen ve aralarında etkileşim bulunan parçaların oluşturduğu bir bütündür. Benzer bir yaklaşımla Senge (1990), sistemi, belirlenmiş bir hedefe ulaşmak amacıyla, aralarında fonksiyonel ilişkiler bulunan fiziksel veya kavramsal bileşenlerin organik bütünlüğü olarak tanımlamaktadır.

Sistemi oluşturan unsurların aritmetik toplamı, sistemin bütününü ifade etmekte yetersiz kalır; sistem, parçaların toplamından daha büyük bir anlam ifade eder (sinerji). Bir sistemin temel karakteristikleri şunlardır:

- Sınırlar: Her sistemin onu çevresinden ayıran bir sınırı vardır.
- Hiyerarşi: Sistem, kendinden daha küçük alt sistemlerden oluşur ve daha büyük bir üst sistemin parçasıdır.
- İlişki ve Bağımlılık: Unsurlar arasında karşılıklı etkileşim mevcuttur.
- Amaç: Tüm bileşenler ortak bir hedefe yönelmiştir.

Sistemler, yapılarına ve işleyişlerine göre çeşitli şekillerde sınıflandırılabilir (Boulding, 1956):

- Açık-Kapalı Sistemler: Çevresiyle madde/enerji/bilgi alışverişinde bulunan (Açık) veya bulunmayan (Kapalı).
- Deterministik-Probabilistik Sistemler: İşleyişi ve çıktısı kesin kurallarla belli olan (Deterministik) veya olasılıklara dayalı değişebilen (Probabilistik).
- Doğal-Yapay Sistemler: Doğada kendiliğinden var olan veya insan tasarımı olan sistemler.

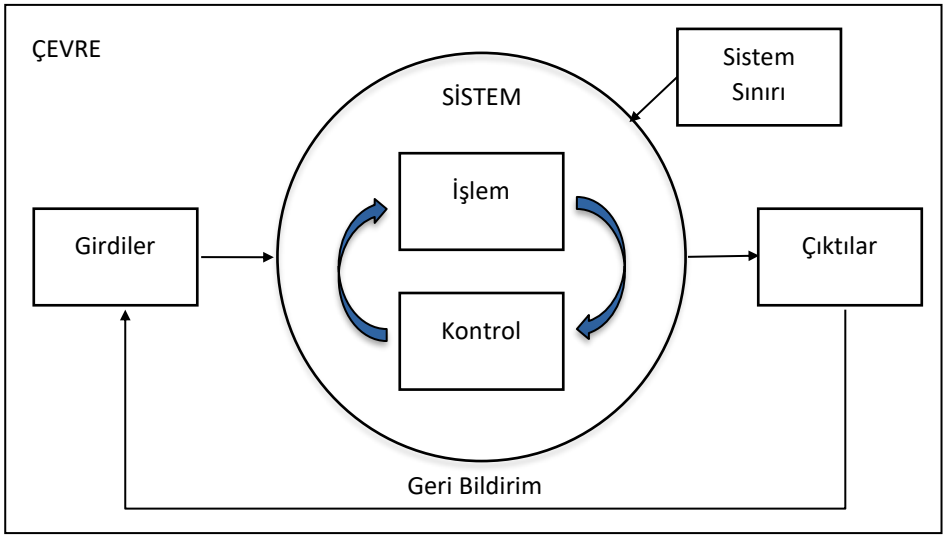
2.1. Bilişim Sistemlerinde Bütüncül (Holistik) Bakış

Bilişim alanında sistem yaklaşımı, sadece kod hatalarını düzeltmeyi değil; hatanın kök nedenine inmeyi, süreçleri ve veri akışlarını analiz etmeyi gerektirir. Bilişimciler, sorunlara bütüncül (holistic) bir bakış açısıyla yaklaşır. Örneğin, bir yazılım hatası sadece bir kod dizim yanlışlığı olmayabilir; girdi aşamasındaki bir kullanıcı arayüzü eksikliği, işlem aşamasındaki veritabanı tutarsızlığı veya çevre faktörü olan ağ gecikmesi sorunun asıl kaynağı olabilir.

Genel Sistem Teorisi (GST), sistemlerin işleyişindeki bu karmaşıklığı yönetmek için temel bir disiplin sunarken; Peter Checkland (1981) tarafından geliştirilen "Yumuşak Sistem Metodolojisi" (Soft Systems Methodology-SSM), insan faktörünün ve belirsizliklerin yoğun olduğu durumlara odaklanır. Geleneksel "Sert Sistemler" (Hard Systems) yaklaşımı, problemleri net tanımlı mühendislik sorunları olarak görürken; SSM, problemleri "insan aktivite sistemleri" olarak ele alır. SSM, farklı paydaşların bakış açılarını analiz etmek için CATWOE (Müşteri, Aktörler, Dönüşüm, Dünya Görüşü, Sahipler, Çevre Kısıtları) gibi araçlar kullanarak, problemin teknik boyutunun ötesindeki sosyal ve yönetimsel dinamikleri de modele dahil eder (Checkland & Scholes, 1990).

2.2. Sistemin İşleyiş Mekanizması ve Döngüsü

Bir sistemin temel dinamikleri; Girdi, İşlem (Süreç), Çıktı ve Geri Bildirim (Feedback) mekanizmalarından oluşur. Şekil 1, bu dinamik döngüyü ve sistemin çevresiyle olan etkileşimini göstermektedir.



Şekil 1: Sistemin Temel İşleyiş Döngüsü

Şekil 1'deki bileşenlerin bilişim sistemleri bağlamındaki karşılıkları şöyledir:

- Girdi: Sisteme giren veri, kaynak ve komutlardır. Bilişimci için girdinin doğruluğu (validation) ve bütünlüğü esastır. Hatalı bir veri girişi, kaçınılmaz olarak hatalı bir sonuca yol açar.
- İşlem: Girdinin çıktıya dönüştüğü mantıksal ve fiziksel adımlardır. Algoritmalar, iş kuralları ve veritabanı sorguları bu aşamada devreye girer.
- Çıktı: Sistemin ürettiği raporlar, arayüz görüntüleri veya diğer sistemlere iletilen verilerdir. Çıktının, kullanıcı ihtiyaçlarını karşılayacak nitelikte olması başarının temel ölçütüdür.
- Geri Bildirim: Sistemin çıktısının, yeniden girdi olarak sisteme dönmesi veya süreci düzenlemek için kullanılmasıdır (Wiener, 1948). Bilişim sistemlerinde log kayıtları, hata mesajları ve kullanıcı deneyimi anketleri, sistemin kendi kendini düzenlemesi (negatif entropi) için kritik öneme sahiptir.
- Sistem Sınırı ve Çevre: Sistemi dış dünyadan ayıran sınırdır. Bir Kurumsal Kaynak Planlaması (ERP) yazılımı için "çevre"; yasal düzenlemeler, rakipler ve tedarikçilerdir. Ackoff'un (1981) belirttiği gibi, kurumlar açık sistemler oldukları için çevresel değişimlere (yeni kanunlar, ekonomik krizler) karşı proaktif bir planlama yapmak zorundadır.

Sistem yaklaşımı, sadece teknik değil, yönetsel ve bireysel gelişim süreçlerinde de uygulanabilir bir metodolojidir. Bir birey, aldığı eğitimi "girdi", zihinsel işlem süreçlerini "süreç", kazandığı yetkinlikleri "çıktı" ve sınav sonuçlarını "geri bildirim" olarak modelleyerek, kendi kariyer gelişimini bir sistem mühendisliği titizliğiyle yönetebilir.

3 | BİLİŞİM SİSTEMLERİ

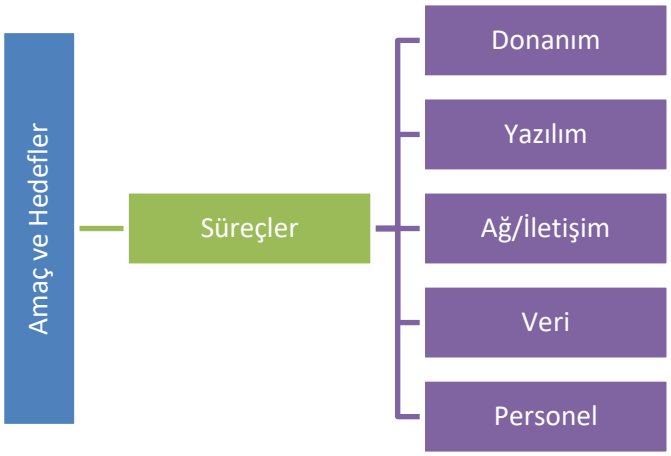
Bilişim sistemleri, günümüz organizasyonlarının sinir sistemini oluşturmaktadır. En temel tanımıyla bir bilişim sistemi; organizasyonel karar verme, koordinasyon ve kontrol süreçlerini desteklemek amacıyla verinin toplanması, işlenmesi, depolanması ve iletilmesini sağlayan, birbiriyle ilişkili bileşenler bütünüdür (Laudon & Laudon, 2020). Bu çalışma, araştırmacılara ve uygulayıcılara bilişim sistemlerinin analizi ve tasarımı konusunda kavramsal bir yol haritası sunmayı hedeflemektedir.

3.1. Bilişim Sistemlerinin Temel Bileşenleri

Bir bilişim sistemi projesinin başarısı, sadece teknolojik yatırıma değil, sistemi oluşturan tüm bileşenlerin uyumlu çalışmasına bağlıdır. Literatürde genellikle donanım ve yazılım ön plana çıkarılsa da, modern yaklaşımda bir bilişim sistemi altı temel bileşenden oluşur: Donanım, Yazılım, Veri, Personel (İnsan), Süreçler (Prosedürler) ve Ağ/İletişim (O'Brien & Marakas, 2011).

Modern literatürde bilişim sistemlerinin altı temel bileşenden (donanım, yazılım, veri, insan, prosedür, ağ) oluştuğu genel kabul görse de; 'Amaç ve Hedefler' bu yapının varlık nedenini oluşturan en kritik unsurdur. Hedefsiz bir sistemin işlevsel olması beklenemeyeceğinden; 'Amaç ve Hedefler', diğer altı bileşeni kapsayan, onlara yön veren ve bütünleştiren kurucu bileşen (üst bileşen) olarak sistemin merkezinde yer almaktadır.

Şekil 2'de gösterilen bu bileşenler arasında "Personel" ve "Veri" unsurları, sistemin başarısı için kritik öneme sahiptir. Araştırmacıların ve proje yöneticilerinin sıklıkla düştüğü hata, projeyi sadece teknik bir "yazılım geliştirme" işi olarak görüp, prosedürleri ve insan faktörünü (kullanıcı adaptasyonu, eğitim vb.) göz ardı etmeleridir. Oysa sosyo-teknik yaklaşım, teknolojinin ancak sosyal sistemle (insan ve süreçler) optimize edildiğinde başarılı olabileceğini savunur.



Şekil 2: Bilişim Sistemi Bileşenleri

3.2. Bilginin Değeri ve Kalite Kriterleri

Sistemin "çıktısı" olan bilginin, organizasyon için bir güç ve değer ifade edebilmesi, belirli kalite standartlarını karşılamasına bağlıdır. DeLone ve McLean'in (2003) Başarı Modeli'nde de vurgulandığı üzere, bilgi kalitesi sistem kullanımını ve kullanıcı memnuniyetini doğrudan etkiler. Nitelikli bir bilginin taşıması gereken temel karakteristikler şunlardır:

- Doğruluk (Accuracy): Bilginin hatasız olması.
- Tamlık (Completeness): Eksik veri barındırmaması.
- Zamanlılık (Timeliness): Karar anında erişilebilir olması.
- İlgililik (Relevance): Karar vericinin ihtiyacına uygun olması.
- Ekonomiklik: Bilgiyi elde etme maliyetinin, sağladığı faydadan düşük olması.

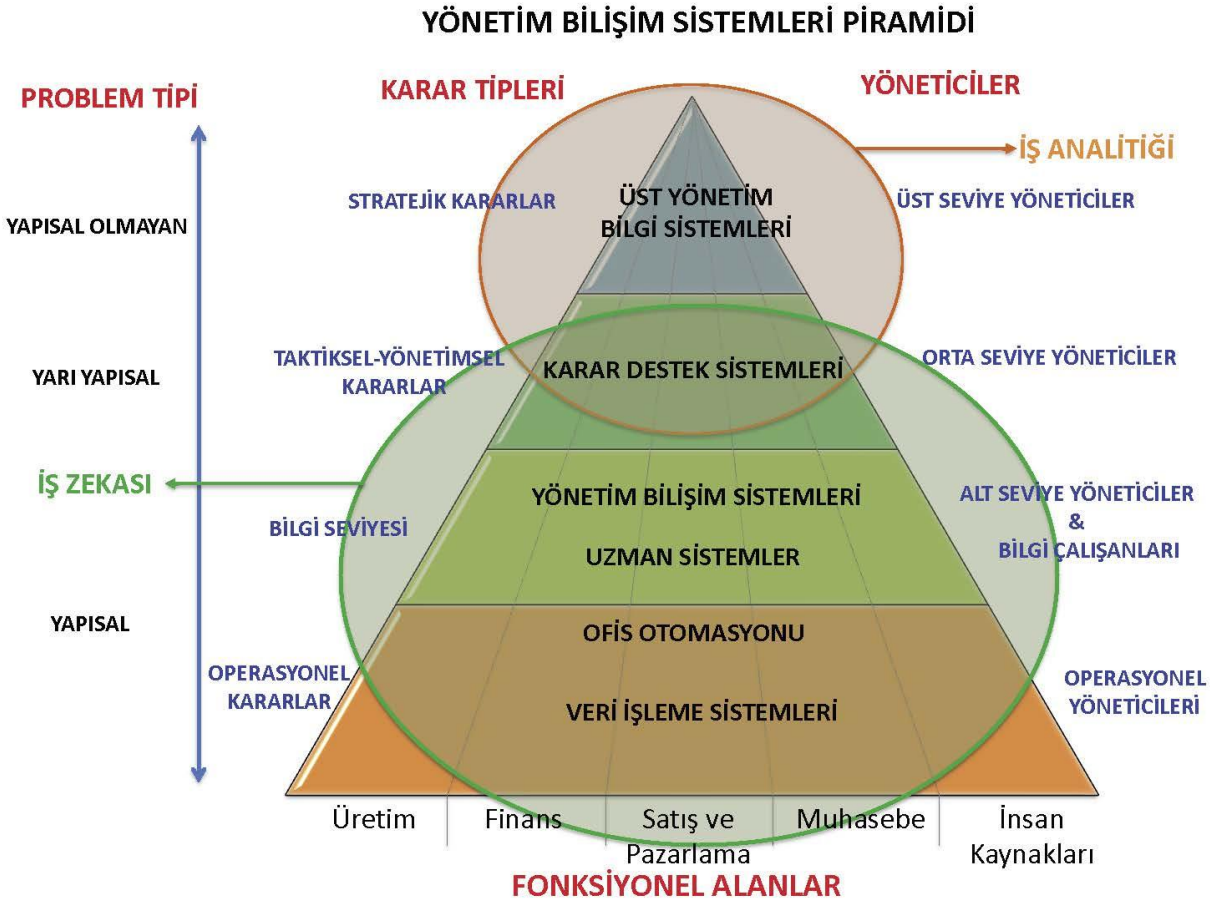
3.3. Proje Yönetimi ve Başarısızlık Nedenleri

Bilişim projelerinde karşılaşılan başarısızlıkların temelinde, teknik yetersizliklerden ziyade; amaç ve hedeflerin net tanımlanmaması, kapsam kayması (scope creep) ve veri yönetimi sorunları yatmaktadır (Standish Group, 2015). Özellikle akademik tezlerde ve projelerde, veri erişilebilirliği (accessibility) ve kullanılabilirliği başlangıçta analiz edilmediğinde (fizibilite eksikliği), süreç içinde yaşanan veri tedarik sorunları projenin hedeflerinden sapmasına veya tamamen başarısız olmasına neden olmaktadır.

Ayrıca teknolojinin baş döndürücü hızı (Moore Yasası ve teknolojik eskime), projeler için büyük bir risk faktörüdür. Uzun süreli planlanan projelerde, başlangıçta seçilen donanım veya yazılım platformu, proje bitmeden güncelliğini yitirebilmektedir. Bu nedenle, aşağıda detaylandırılacak olan Sistem Geliştirme Yaşam Döngüsü (SDLC), bu riskleri minimize etmek için disiplinli bir süreç yönetimi sunar.

3.4. Yönetim Bilişim Sistemleri (YBS) Piramidi

Bir organizasyonda bilişim sistemleri, hizmet ettikleri yönetim kademesine ve çözdükleri problemin yapısına göre sınıflandırılır. Robert Anthony (1965) tarafından geliştirilen ve Şekil 3'te görselleştirilen YBS Piramidi, bu hiyerarşik yapıyı açıklar (Tecim, 2025a).



Şekil 3: Analitik Yetkinlik Tabanlı YBS Piramidi

Piramit incelendiğinde üç temel seviye görülmektedir:

1. **Operasyonel Seviye (En Alt):** Günlük rutin işlerin (satış, bordro, stok) takip edildiği, yapısal problemlerin çözüldüğü ve "Veri İşleme Sistemleri" (veya Süreç Yönetim Sistemleri), "Uzman Sistemler" ve "Yönetim Bilişim Sistemleri"nin kullanıldığı seviyedir.
2. **Taktiksel/Yönetimsel Seviye (Orta):** Orta düzey yöneticilerin performans izleme ve denetim faaliyetlerini yürüttüğü, yarı yapısal problemlerin olduğu ve "Karar Destek Sistemleri (KDS)"nin kullanıldığı alandır.
3. **Stratejik Seviye (En Üst):** Üst düzey yöneticilerin uzun vadeli planlamalar yaptığı, belirsizliğin yüksek olduğu ve dış çevre verisine ihtiyaç duyulan, yapısal olmayan problemlerin "Üst Yönetim Bilgi Sistemleri" (veya Üst Seviye Destek Sistemleri) ile çözüldüğü seviyedir.

Sistem analisti, hangi seviye için geliştirme yapacağını netleştirerek; o seviyenin veri ihtiyacına (detaylı vs. özet) ve problem yapısına uygun metodolojiyi belirlemelidir.

3.5. İş Zekası ve İş Analitiği ile YBS Entegrasyonu

İş Zekası (Business Intelligence - BI) ve İş Analitiği (Business Analytics - BA) kavramları, geleneksel piramidin modern versiyonlarında "Yönetim Bilişim Sistemleri" ve "Karar Destek Sistemleri" katmanlarının evrimleşmiş hali veya tamamlayıcısı olarak konumlandırılmalıdır.

İş Zekası (İZ) genellikle "Ne oldu?" ve "Neden oldu?" sorularına cevap verir, tarihsel veriyi raporlar. Bu nedenle geleneksel Veri İşleme Sistemleri (VİS) ile YBS katmanını kapsar ve KDS katmanının alt sınırına dokunarak tanımlayıcı analitik işlevini görür (Şekil 3).

İş Analitiği (İA) tahminleme amaçlı "Ne olacak?" ve optimizasyon amaçlı "Ne yapmalıyız?" sorularına cevap verir, istatistiksel modelleme ve yapay zeka içerir. Bu nedenle İş Analitiği KDS ve Üst Yönetim Bilgi Sistemleri katmanlarını kapsar. Burada amaç orta ve üst düzey yöneticilere öngörücü ve reçete verici analitik işlemlerin sonucunu sunmaktır. Stratejik kararlar iş analitiğin en yoğun kullanıldığı alandır.

Geleneksel YBS Piramidi'nde yer alan VİS, YBS ve KDS günümüz veri odaklı organizasyonlarında evrilmiştir. Bu bağlamda, tarihsel verilerin raporlanmasını sağlayan YBS katmanı *İş Zekası* şemsiyesi altında; geleceğe yönelik tahminleme ve simülasyonların yapıldığı KDS katmanı ise *İş Analitiği* yetenekleri ile zenginleşerek piramidin üst katmanlarına entegre olmuştur (Davenport, 2006).

Şekil 3'de görüldüğü üzere KDS hem İş Zekası hem de İş Analitiği kümesi içinde yer almaktadır. Çünkü KDS, hem geçmiş raporlara (İZ) hem de gelecek simülasyonlarına (İA) ihtiyaç duyar. Bu grafik artık klasik bir YBS piramidi olmaktan çıkıp, modern veri analitiği süreçlerini de içeren hibrit bir modele dönüşmüş durumda.

4 | SİSTEM GELİŞTİRME YAŞAM DÖNGÜSÜ

Sistem Geliştirme Yaşam Döngüsü (SGYD); girdilerin ve etken unsurların ölçülebildiği, sınırları net olarak tanımlanmış problemlerin çözümünde kullanılan ve hata payını minimize etmeyi amaçlayan sistematik bir süreçtir (Valacich ve George, 2020). Temel amacı; karmaşık bir problemi, ölçülebilir çıkış kriterleri ile birbirine bağlanan, iyi tanımlanmış aşamalar setine bölerek yönetilebilir hale getirmektir. Bu bağlamda SGYD, modern sistem analizi ve tasarım metodolojilerinin temelini oluşturmaktadır (Kendall ve Kendall, 2019).

Bir metodolojinin, araştırmacılar veya uygulayıcılar tarafından takip edilen prosedürler ve kurallar bütünü olduğu düşünüldüğünde (Avison ve Fitzgerald, 2006); SGYD disiplini dayatarak bir bellek desteği işlevi görür ve süreç boyunca kritik detayların göz ardı edilme riskini azaltır. Metodoloji kullanımı, sağladığı standart belge ve iletişim normları sayesinde programlama, bütçeleme ve zaman planlaması gibi süreçlerde yönetimsel kontrolü güçlendirir (Whitten ve Bentley, 2007). Davis'in (1999) de belirttiği üzere; iyi yapılandırılmış bir metodoloji, proje araçlarının etkin kullanımıyla birleştiğinde, önemli hataların erken evrelerde tespit edilme olasılığını artırmaktadır.

Ancak, metodoloji kullanımının getirdiği disiplin ile esneklik arasındaki dengenin gözetilmesi kritik önem taşımaktadır. Süreç odaklılığın sonuç odaklılığın önüne geçmesi, yani sadece adımları tamamlama gayretinin asıl problemin çözümünü gölgelemesi, metodolojilerin uygulanmasındaki temel risklerden biridir (Dennis vd., 2015). Hangi metodoloji seçilirse seçilsin, yöntemin problemin doğasına uygun olması gerekmektedir. Örneğin, bazı geleneksel yöntemler; uzman sistemler, gerçek zamanlı işlem sistemleri veya nesne yönelimli dillerin gerektirdiği dördüncü nesil yaklaşımlar için yetersiz kalabilmektedir (Sommerville, 2011).

Teknolojideki hızlı gelişmelere rağmen vurgulanması gereken husus; hiçbir metodolojinin veya teknolojik aracın, yetkin bir analistin yerini alamayacağıdır. İyi bir metodoloji, vasıfsız bir kişiyi uzmana dönüştürmez; ancak yetkin bir analistin verimliliğini artırarak onu daha üretken kılar. Sonuç olarak SGYD, karmaşık görevler için kademeli ve yapısal bir çerçeve sunmakla birlikte; uygulamada analistlerin projenin gerekliliklerine göre bu katı yaklaşımları esneterek uyarlamalar yaptıkları da görülmektedir (Hoffer vd., 2017).

SGYD, sistem analistlerinin ve kullanıcı faaliyetlerinin spesifik döngüsünün kullanılmasıyla en iyi geliştirilmiş sistemin oluşmasını sağlayan analiz ve tasarım için aşamalı bir yaklaşımdır (Tecim, 2025b). SGYD, bir bilişim projesinin doğumundan ölümüne kadar geçen sürecin haritasıdır.

SGYD'nü, birbirinden ayrı adımlar olarak değil, safhalar içinde başarılan bir süreç olarak düşünmek daha faydalı olacaktır.

Şekil 4: Sistem Geliştirme Yaşam Döngüsü Aşamaları

Her bir aşama detaylı olarak ele alınıp, yapılması gereken işlemler anlatılacaktır.

1. Problemlerin, Fırsatların ve Amaçların Tanımlanması

Sistem Geliştirme Yaşam Döngüsü'nün (SDLC) en stratejik ve kritik aşaması olan bu safha, projenin teorik ve pratik temellerinin atıldığı yerdir. Literatürde, yazılım hatalarının maliyeti üzerine yapılan çalışmalarda (Boehm, 1981), gereksinim ve tanımlama aşamasında yapılan bir hatanın, bakım aşamasında düzeltilmesinin 100 kat daha maliyetli olduğu belirtilmiştir. Halk deyişiyle, eğer bu aşamada düğme yanlış iliklenirse, sonraki adımlar ne kadar mükemmel icra edilirse edilsin, sonuç başarısız olacaktır.

Bu aşama, metodolojik olarak bir teşhis (diagnosis) evresidir. Sistemin ana aktörü olan sistem analisti, tıpkı bir tıp doktoru gibi hareket etmelidir. Dennis, Wixom ve Tegarden'ın (2015) belirttiği üzere, analist tedaviye (kodlama/tasarım) başlamadan önce hastanın şikayetini (problem), vücudun potansiyelini (fırsat) ve tedaviden beklentiyi (amaç) netleştirmelidir.

Problemlerin tanımlanması sürecinde mevcut durumu değişime zorlayan unsurlar, Wetherbe'nin PIECES çerçevesi (Performans, Bilgi, Ekonomi, Kontrol, Verimlilik, Hizmet) ışığında analiz edilmelidir. İşletmede süreçler yavaş mı? Hata oranı veya maliyetler yüksek mi? Burada kritik olan, "belirti (symptom)" ile "kök neden (root cause)" ayrımını yapabilmektir. Örneğin; "Satışlar düştü" bir belirtidir. "Stok takibi yapılamadığı için ürün yok satıyoruz" ise belirtiyi doğuran gerçek ve önemli bir sorundur (Kendall & Kendall, 2019).

Fırsatların belirlenmesi aşaması yalnızca sorunları onarmakla sınırlı kalmamalı; işletmeyi rekabetçi kılacak fırsatları da kapsamalıdır. Yapay zeka, bulut bilişim veya mobil entegrasyon gibi yıkıcı teknolojilerin (disruptive technologies) işletmeye sağlayacağı "Rekabet Avantajı" (Porter, 1985) araştırılmalıdır. Analist, "Rakipler ne yapıyor ve biz teknolojiyi kullanarak onların önüne nasıl geçeriz?" sorusuna yanıt aramalıdır. Fırsatlar, bilgi sistemlerinin stratejik kullanımıyla işletmenin pazar konumunu güçlendireceği alanlardır.

Problemler ve fırsatlar ışığında, hedeflenen durum somut amaçlara dönüştürülmelidir. Amaçlar, SMART (Ölçülebilir, Ulaşılabilir, Zamanlı vb.) kriterlerine uygun olarak ifade edilmelidir. İşletme; "Bu sorunu çözersek X kadar kâr edeceğiz" veya "İşlem süresini Y dakikaya düşüreceğiz" şeklinde, projenin yaratacağı iş değerini somutlaştırmalıdır.

Bu süreç, analistin masa başında tamamlayabileceği bir süreç değildir; ampirik bir saha çalışması gerektirir. Valacich ve George (2017), başarılı bir analiz için "Kullanıcı Katılımı"nın şart olduğunu vurgular. Analist, sadece yöneticilerle değil, işi fiilen yürüten personelle de görüşmelidir. Burada Peter Checkland'in "Yumuşak Sistem Metodolojisi"nde (SSM) vurguladığı Weltanschauung (Dünya Görüşü) kavramı devreye girer. Farklı paydaşlar aynı olayı farklı yorumlayabilir:

Yönetici kendi makro perspektifinden "Sistem yavaş" diyebilir.

Ancak operasyonu yürüten personel "Sistem yavaş değil, ekran tasarımı çok karmaşık olduğu için verileri giremiyorum" diyebilir. Gerçeğin bu iki farklı perspektifin analiz edilmesiyle ortaya çıkarılması gerekir. Analist; iş akışını yerinde gözlemleyerek (gözlem tekniği), kağıtların nerede biriktiğini, çıktıların nereden alındığını ve darboğazların nerede oluştuğunu tespit etmelidir.

Tanımlama safhasının en önemli çıktısı, projenin "yapılabilirliğini" ortaya koyan Fizibilite Raporudur. Bu rapor üç temel boyutta ele alınmalıdır (O'Brien & Marakas, 2011):

1. Teknik Fizibilite: Donanım ve yazılım yetkinliği yeterli mi?
2. Ekonomik Fizibilite: Yapılacak yatırımın geri dönüşü (ROI) tatmin edici mi? Yani halk tabiriyle; "Atılan taş, ürkütülen kurbağaya değer mi?"
3. Operasyonel Fizibilite: Önerilen çözüm, kurum kültürü ve çalışanlar tarafından benimsenebilir mi?

Yönetim, bu rapordaki bulgulara dayanarak "Devam Et" veya "Durdur" kararını verir.

Geleneksel yöntemlerle çalışan, mağazalardan siparişlerini telefonla veya WhatsApp ile alan, kağıda not alıp bunu depoya ileten ve depodaki görevlinin kağıda bakıp ürünü hazırlayan bir toptan gıda işletmesini ele alarak bu aşamanın içeriği daha net ve somut olarak ortaya konulacaktır.

Mevcut durumun vahametini belirten sıkıntılı noktalar ortaya konularak problemler tespit edilir. Telefonla alınan siparişlerde yanlış ürün not ediliyor (%15 oranında iade var) ve hata oranı belirlenir. Sipariş kağıtları depoda kayboluyor, teslimatlar aksıyor durumu ile veri kaybı yaşanıyor. Satış ekibi depoda mal var sanıp sipariş alıyor, ancak depo boş. Bu stok uyuşmazlığı müşteriye karşı mahcubiyet yaşatıyor. Bir siparişin alınması ve faturaya dönüşmesi ortalama 45 dakika sürüyor ve bu zaman kaybına neden oluyor. Hata oranı, veri kaybı, stok uyuşmazlığı ve zaman kaybı tespit edilen problemler olarak belirlenmiştir.

Problemlerin tespitinin ardından fırsatların belirlenmesi için teknolojinin nasıl bir kaldıraç olacağı ortaya konulmalı. Saha satış elemanlarına tablet verilirse sipariş anında sisteme düşebilir (mobil sipariş). Depo ile satış sistemi entegre edilirse, olmayan ürünün satışı engellenir (anlık stok takibi). Hızlı ve hatasız teslimat sayesinde rakiplerden müşteri çalınabilir (müşteri sadakati). Hızlı işeyen süreç sonunda anında fatura kesilmektedir (işletmeye güven)

Problemlerin ve fırsatların belirlenmesinden sonra amaçların net olarak ortaya konması ile proje hedeflerin belirlenmiş olur. İade oranını %15'ten %1'in altına indirmek (hedef 1), sipariş işleme süresini 45 dakikadan 2 dakikaya düşürmek (hedef 2) ve stok verisini %100 doğrulukla, canlı olarak takip edilebilir hale getirmek (hedef 3) proje için belirlenebilecek hedefler olabilir.

Birinci aşama kod yazmak veya sunucu kurmak değil, ”neyi, neden yapıyoruz ve bu işe girişmeye değer mi?” sorusunun cevabını, rakamlarla ve gerçekçi verilerle ortaya koyma sanatıdır.

2. Bilgi Gereksinimlerinin Belirlenmesi

Projenin bu aşaması, analistin bir dedektif titizliğiyle çalıştığı, ampirik veri toplama sürecidir. İlk aşamada belirlenen "neyin" yapılacağı sorusundan sonra, bu aşama sistemin "içinin neyle ve nasıl dolacağının" kurgulandığı safhadır. Literatürde Davis (1982) tarafından belirtildiği üzere, bilgi gereksinimlerini doğru tespit etmek, sistem başarısızlığını önleyen en kritik faktördür. Analistin temel amacı, yazılımcı ve tasarımcıların önüne koyulacak teknik şartnameyi, yani metaforik olarak "yemek tarifinin malzemelerini" eksiksiz hazırlamaktır.

Analist, gereksinimleri toplarken sistematik bir sorgulama yöntemi izlemelidir. Bu noktada Zachman Çerçevesi (Zachman Framework), analiste rehberlik eder (Zachman, 1987). Analist, sistemin en ince detaylarını ortaya çıkarmak için şu boyutları irdelemelidir:

- Kim (Who): Sistemi kullanacak aktörler ve paydaşlar kimlerdir?
- Ne (What): İşlenen veri ve varlıklar nelerdir?
- Nerede (Where): İşletmenin coğrafi dağılımı ve ağ altyapısı nasıldır?
- Ne Zaman (When): Süreçlerin zamanlaması ve tetikleyicileri nelerdir?
- Nasıl (How): Süreçlerin işleyiş mantığı ve yöntemleri nasıldır?
- Neden (Why): İşletme neden mevcut yöntemi kullanıyor ve değişim amacı nedir?

Mevcut sistemin (Legacy System) kullanımına devam edilmesinin geçerli sebepleri olabilir; ancak analist, işletme fonksiyonlarını, insan faktörünü ve veri akışını tam olarak anlayarak daha verimli bir tasarım önermelidir.

Kendall ve Kendall (2019), bilgi ihtiyaçlarının belirlenmesinde üç temel yöntemin altını çizer:

1. Etkileşimli Yöntemler: Mülakatlar, anketler ve JAD (Ortak Uygulama Tasarımı) oturumları.
2. Müdahalesiz Yöntemler: Ham verinin örneklenmesi (Sampling), belge incelemesi ve ofis çevresinin gözlemlenmesi (Observation).
3. Prototipleme ve RAD: James Martin (1991) tarafından geliştirilen Hızlı Uygulama Geliştirme (Rapid Application Development - RAD) yaklaşımı, kullanıcıların ihtiyaçlarını teorik olarak anlatmak yerine somut bir prototip üzerinden görmelerini sağlar. Bu, özellikle belirsizliğin yüksek olduğu durumlarda nesne yönelimli ve iteratif bir çözüm sunar.

Mevcut sistemin kullanılması için geçerli sebepler olabilir, ancak daha iyi bir sistem dizaynı düşünülmelidir. Analist, işletme fonksiyonlarını anlamalı ve insanların, hedeflerin, verilerin ve kullanılan yöntemlerin tam bilgisine sahip olmalıdır.

Aynı örnek ele alındığında sistem analisti depoya, muhasebeye ve sahaya inerek kullanıcılarla konuşarak (mülakatlar), kurumdaki mevcut belgeleri inceleyerek ve gözlemleyerek bilgiler toplar.

Sistemin paydaşlarıyla mülakatlar yapılır. Saha satış elemanına "Siparişi alırken müşterinin hangi bilgilerine ihtiyacın oluyor?" sorusuna alınacak "Müşterinin borcu var mı görmem lazım. Eğer borcu limitin üstündeyse sistem bana 'Satış Yapma' uyarısı vermeli. Ayrıca o anki güncel stok miktarını görmezsem yine olmayan ürünü satarım." cevabı ile yaratılacak sistemin, canlı stok verisini ve müşterinin cari bakiye bilgisini anlık çekmesi gerektiği belirlenmiş olur. Depo sorumlusu ile yapılan görüşmede "Kağıt sipariş fişlerinde en çok neyde zorlanıyorsunuz?" sorusuna alınacak "Satışçılar ürün kodunu yazmıyor, sadece ismini yazıyor (örn: 'Domates Salçası'). Ama bizde 830 gramlık da var, 5 kiloluk da. Hangisi olduğunu anlamak için aramak zorunda kalıyorum." cevabı ile oluşturulacak sistemin ürün seçimi manuel yazarak değil, barkod okutarak veya listeden fotoğraf seçilerek yapılması, gramaj bilgisini zorunlu olarak alması gerektiği belirlenir.

İşletmede şu an kullanılan manuel sipariş koçanları, faturaları ve varsa Excel tabloları incelenir. Eski kağıt formlarda "Teslimat Tarihi" diye bir alan olduğunu ama çoğunun boş bırakıldığı, atış elemanlarına "Bunu neden doldurmuyorsunuz?" diye sorulduğunda "Müşteri genelde 'hemen gelsin' diyor, tarih belirtmiyor" cevabını alındığı tespit edilir. Bu durumda kurulacak sistemde varsayılan teslimat tarihi "Yarın" olarak ayarlanmalı ama değiştirilebilir olmalıdır olarak belirlenebilir.

Analisti, bir gününü saha satış elemanıyla araçta geçirerek elemanın bir markete girdiğinde internetin çekmediğini (bodrum kat depoları vs.) gözlemler ve geliştirilecek uygulamanın çevrimdışı çalışabilmesi, siparişleri internetsiz ortamda kaydedip, interneti görünce merkeze göndermesi gerektiği belirlenir.

Bu bilgiler toplandıktan sonra analist kimin neyi ne zaman yapması gerektiğini yani iş sürecini; veritabanının oluşturulması için müşteri, ürün ve sipariş verilerinin neler olacağını; sistemin kısıtlarının ve kurallarının (stokta olmayan ürün sepete eklenmemeli, toplam tutar 1000 TL altındaysa sipariş onaylanmamalı, saat 17:00'dan sonra girilen siparişler "Ertesi Gün" sevkiyatına düşemez, "Sonraki Gün"e sarkar şeklinde) neler olacağı belirlemiş oluyor.

Bu aşama hayali, gerçeğe dönüştürecek tüm detayların masaya yatırıldığı adımdır. George'un (2017) belirttiği gibi, kullanılabilirlik (usability) testlerinin temeli bu sahada atılır. Büyük emeklerle ve harcamalarla ortaya çıkacak ürünün son kullanıcı tarafından kullanılabilirliğine temel oluşturacak unsurlar sahada tespit edilmedikçe ürünün kullanması mümkün olamayacaktır.

3. Sistem İhtiyaçlarının Analizi

Bu aşama, ikinci safhada mülakatlar, gözlemler ve belge incelemeleri yoluyla elde edilen ham verilerin; mantıksal bir düzene oturtulduğu ve "sistematikleştirildiği" evredir. Literatürde "Gereksinim Mühendisliği" (Requirements Engineering) olarak da adlandırılan bu süreçte; dağınık haldeki kullanıcı istekleri, iş kuralları ve süreçler, teknik bir sistem mimarisine dönüştürülür (Sommerville, 2011). Amaç, mevcut durumun (As-Is) fotoğrafından yola çıkarak hedeflenen sistemin mantıksal modelini kurmaktır.

Karmaşık iş süreçlerini anlaşılır kılmak için analist spesifik modelleme araçlarından faydalanır. Bu araçların başında, Gane & Sarson (1979) veya Yourdon metodolojisiyle ilişkilendirilen Veri Akış Diyagramları (Data Flow

Diagrams - DFD) gelir. DFD; işletme fonksiyonlarını girdi, süreç (process) ve çıktı bağlamında görselleştirir. Diyagramlardaki her bir veri hareketinin ve depolanan verinin tanımlandığı yapıya ise Veri Sözlüğü (Data Dictionary) denir. Veri sözlüğü, sistemdeki tüm veri elemanlarının "meta verisini" (verinin verisi) barındırarak, yazılım ekibi ile iş birimleri arasında ortak bir dil oluşturur (Kendall & Kendall, 2019).

Sistem analisti, sadece veri akışını değil, işletmenin karar verme mekanizmalarını da analiz etmelidir. Herbert Simon'ın (1960) karar teorisine göre kararlar iki ana sınıfta incelenir:

Yapılandırılmış Kararlar (Programlanabilir): Kuralları, şartları ve sonuçları belli olan kararlardır. Analist bu kararları modellemek için Karar Tabloları, Karar Ağaçları veya Yapılandırılmış Dil kullanır.

Yarı-Yapılandırılmış ve Yapılandırılmamış Kararlar: Belirsizlik içeren ve yöneticinin yargısına dayanan bu kararlar (örneğin risk altındaki kararlar), genellikle Karar Destek Sistemleri (DSS) kapsamına girer. Burada analist, Çok Kriterli Karar Verme (MCDM) tekniklerini kullanarak problemin karmaşıklığını ve ağırlıklandırma kriterlerini modeller.

Analiz safhasının sonunda, alternatif çözüm yollarının fayda/maliyet analizini içeren bir Sistem Önerisi hazırlanır. Unutulmamalıdır ki; sistem teorisinde "Eşsonluluk" (Equifinality) ilkesi gereği, bir problemin tek bir doğru çözümü yoktur (Bertalanffy, 1968). Çözüm, analistin yetkinliğine ve kurumun kısıtlarına göre şekillenir.

Bu aşamada yapılacak işlemler veri akışının şemalaştırılması, karar mekanizmalarının belirlenmesi ve fonksiyonel gereksinimlerin belirlenmesi başlıkları ile net olarak ortaya konulabilir.

Aynı örnek üzerinden gidilecek olursa veri akışının şemalaştırılmasında (DFD- Data Flow Diagrams analist işletmedeki veri trafiğinin haritasını çizerek kod yazmayı değil mantıksal akışı ortaya koyar. İşletmedeki bir siparişin yolculuğunun şemasını çizer. Girdi olarak satış elemanı siparişi girer, sistem "müşteri borç limiti" veritabanını giderek müşteriyi sorgular ve limit durumuna göre satış elemanına uygunsuzluk veya satış onayı karar olarak depo stok sistemine miktar olarak düşer ve oradan depo sorumlusuna ürün hazırlama fişi çıkar. Ortaya çıkan bu şemaya uygun olarak veritabanında varlık ilişki (ER-Entity Relationship) diyagramı oluşturularak tablolar arası ilişkiler belirlenir.

İşletmede belirlenmiş olan kuralların yer aldığı "Eğer-Öyleyse" (If-Else) mantığının yer aldığı süreçte örnek işletmede "Müşterinin borcu varsa satış yapma" kuralı için analiz yapılarak karar tablosu oluşturulur. Bu süreç duruma göre ya "Borcu var ama 30 günü geçmemiş -> Satışa İzin Ver", ya "Borcu yok ama stokta ürün yok -> Siparişi Al, "Bekleyen"e at" ya da "Borcu limiti aşmış ve 60 günü geçmiş -> Sistemi Kilitle, Genel Müdüre Bildirim Gönder." şeklinde karara dönüşür. Depocunun inisiyatifine bırakılan kurallar, artık sistemin katı kuralları haline gelmiş olacaktır.

Analist, toplanan istekleri teknik bir dille fonksiyonel ve fonksiyonel olmayan gereksinimler olarak iki başlık altında toplayarak yazılımcıya rehber olacak şekilde doküman haline getirmelidir. Fonksiyonel gereksinimlerde kurulması planlanan yeni "sistem ne yapmalı" sorusunun cevabı aranmalı. Örnek işletme ele alındığında yeni sistem, barkod okuyucu ile ürün ekleyebilmeli, internet kesildiğinde veriyi tablette tutmalı, internet gelince merkeze yollamalı, her akşam saat 19:00'da satış temsilcisi bazlı satış raporu üretmeli, muhasebe programı (Logo/Netsis/ vb.) ile entegre çalışmalıdır gibi kurallar yer almalı. Fonksiyonel olmayan gereksinimlerde kurulması planlanan yeni "sistem nasıl çalışmalı" sorusunun cevabı aranmalı. Örnek işletmede ortaya çıkacak uygulamada, bir ürün aratıldığında sonuç en geç 2 saniye içinde ekrana gelmeli, satış elemanları sadece kendi müşterilerini görmeli, diğer bölgeyi görememeli, ekran butonları depoda eldivenle çalışanlar için büyük olmalı ve arayüz sade olmalı, piyasadaki uygun fiyatlı Android tabletlerde (min. 4GB RAM) çalışabilmeli şeklindeki kurallar ile performans, güvenlik, kullanılabilirlik ve donanım gereksinimleri belirlenmelidir.

Bu aşamada analist yönetime ve yazılım ekibine kapsamlı bir dosya sunarak sistemin mantıksal tasarımını ortaya koyar. Sözün bittiği, mantığın ve kuralların kağıda döküldüğü bu aşamadan sonra gelen tasarım aşaması tamamen bu aşamada yer alan detaylı dokümanlara bakarak çizilebilecektir. Bu aşamanın atlanması veya doğru bir şekilde yapılmaması durumunda yazılımcı "Borcu olan müşteriye ne yapayım?" diye her defasında sormak zorunda kalır veya "Sistem çok yavaş çalışıyor" şikayeti gelir çünkü fonksiyonel olmayan gereksinimde yer alan performans kriteri başta belirlenmemiştir.

4. Önerilen Sistemin Tasarımı

Bu aşama, analiz evresinde belirlenen "ne yapılmalı" (gereksinimler) sorusunun, teknik ve mimari açıdan "nasıl yapılmalı" sorusuna dönüştürüldüğü kritik bir geçiş evresidir. Whitten ve Bentley'in (2007) tanımlamasıyla tasarım; henüz kod yazılmadan sistemin her detayının kağıt üzerinde veya dijital modelleme araçlarıyla (CASE

tools) "inşa edildiği" mühendislik sürecidir. Amaç, yazılım geliştirme sürecindeki belirsizlikleri minimize etmektir.

Analist, hatasız veri-giriş prosedürünü tasarlar. Böylece bilgi sistemine giren verinin doğruluğu sağlanır. İyi bir form ve ekran tasarım tekniği kullanarak bilgi sistemi için etkili veriler sağlar.

Bilgi sisteminin mantıksal tasarımın bir parçası olarak kullanıcı arayüzü oluşturulur. Arayüz, kullanıcı ile sistem arasındaki "iletişim köprüsüdür". Shneiderman'ın (2016) "Arayüz Tasarımının 8 Altın Kuralı" ışığında, kullanıcı hatalarını önleyen ve veri girişini kolaylaştıran ekranlar tasarlanır. Bu aşama, işletmede karar vericilerin ihtiyaç duyduğu çok miktardaki veriyi depolayacak dosyaların ve veri tabanının tasarımını içerir.

İyi düzenlenmiş bir veri tabanı tüm bilgi sistemlerinin temelidir. Analist, sistemi ve veriyi korumak için ve programcılara programı geliştirebilmeleri için kontrol ve geri-dönüş prosedürleri tasarlamalıdır.

Bu aşamada mantıksal ve fiziksel tasarımlar yapılır. Sisteme ait tüm bileşenlerin, donanımdan bağımsız olarak, sadece işlevsellik açısından tasarlandığı mantıksal tasarımda girdi, çıktı ve veritabanı tasarımları yapılmaktadır. Girdi tasarımında kullanıcının sisteme veri gireceği tüm ekranlar ve formlar tek tek tasarlanır. Örnek işletme ele alındığında saha satış elemanının sipariş girmesi gereksinim olarak belirlendiğinde mobil uygulamada müşteri seçme, ürün sepeti ve siparişi onayla butonunun konumu çizilir. Bu tasarımda, depodaki elemanın eldivenle dahi kolayca basabileceği büyük butonlar ve basit arayüz kuralları dikkate alınır.

Çıktı tasarımında sistemden elde edilecek tüm raporlar ve ekran çıktıları planlanmalıdır. Yöneticinin akşam 19:00'da satış temsilcisi bazlı satış raporu alması gereksinimine göre bu raporun hangi alanları (Satış Temsilcisi Adı, Toplam Satış Miktarı, İade Oranı, Yeni Müşteri Sayısı) içereceği ve hangi formatta (PDF, Excel, mobil uygulama paneli) sunulacağı tasarlanır.

Bir önceki aşamada belirlenen varlık ilişki (ER) diyagramına göre "Müşteri" tablosu, "Sipariş" tablosu, "Ürün" tablosu gibi varlıklar belirlenir ve veritabanı şeması oluşturulur. Bu tabloların birbirleriyle nasıl ilişkileneceği (örneğin, bir Müşteri birden çok Sipariş verebilir) netleştirilir.

Fiziksel tasarımda sistemin donanım, yazılım ve ağ altyapısı ile ilgili kararlar alınır. Gereksinim olarak belirlenen mobil uygulamanın çevrimdışı çalışabilmesi koşuluna uygun olarak mobil uygulamanın Android işletim sistemini kullanmasına karar verilir. Verilerin merkezde tutulması için bir Sunucu (Server) ve Veritabanı Yönetim Sistemi (SQL Server veya PostgreSQL gibi) seçilir. Muhasebe programı (Logo/Netsis vb.) ile nasıl bir iletişim kurulacağı (API/Web Servis) yani entegrasyon belirlenir. Ağ ve güvenlik açısından satış elemanlarının mobil uygulaması ile merkez sunucusu arasındaki veri alışverişinin SSL şifrelemesi (HTTPS) kullanılarak yapılmasına karar verilir. Sisteme kimlerin hangi yetkilerle gireceği (Rol ve Yetki Matrisi) detaylandırılır.

Bu aşamada iş akışı ve prosedür tasarımı da yapılarak yeni sistemin devreye girmesiyle değişen kullanıcı görevleri (prosedürler) belgelenir. Yeni sistemde depo görevlisi, kağıt fiş yerine mobil uygulamadan gelen "Hazırlama Fişini" barkod okutarak onaylayacaktır. Bu yeni süreç yazılı olarak belgelenir.

Sonuç olarak tasarım aşamasında tüm kullanıcı ekranlarının görsellerinin tasarımı, tablolar arası ilişkilerin şeması, programlama dili (Python/Java), veritabanı (SQL), sunucu (Cloud/On-Premise) ile ilgili teknolojik tercihler ve sistemin 6. Adımda neye göre test edileceğinin (örnek işletmede çevrimdışı sipariş 30 saniye içinde senkronize olmalıdır gibi) belirlenmesi aşamasıdır. Tasarımdaki bir hatanın, kodlamada yüzlerce saatlik zaman kaybına neden olabilmesi nedeniyle tüm planların en küçük detaya kadar onaylanmasını sağlandığı bir aşamadır.

Yazılım mühendisliği ekonomisinde (Boehm, 1981) belirtildiği üzere; tasarım aşamasında tespit edilen bir hata kağıt üzerinde silgiyle düzeltilebilirken, kodlama aşamasında yüzlerce saatlik maliyete neden olabilir. Bu nedenle tasarım, kodlamaya geçmeden önceki son "onay" kapısıdır.

5. Yazılımın Geliştirilmesi ve Belge Oluşturulması

Şu ana kadar yapılan her şey genelde kağıt üzerindeydi; planlar, analizler, tasarımlar. Bu aşamada soyut tasarımlar, çalışan bir koda somut bir ürüne dönüşür. Pressman'ın (2014) ifade ettiği üzere bu aşama sadece kod yazmak değil; yazılımın sürdürülebilirliğini sağlayan dokümantasyonun üretildiği ve sistemin kurumsal hafızasının oluşturulduğu süreçtir.

Bu safhada analist, hedeflenen yazılım ürününü ortaya çıkarmak için programcılarla eşgüdümlü çalışır. Kodlama devam ederken, sistemin sürdürülebilirliği için hayati öneme sahip olan Kullanıcı ve Sistem

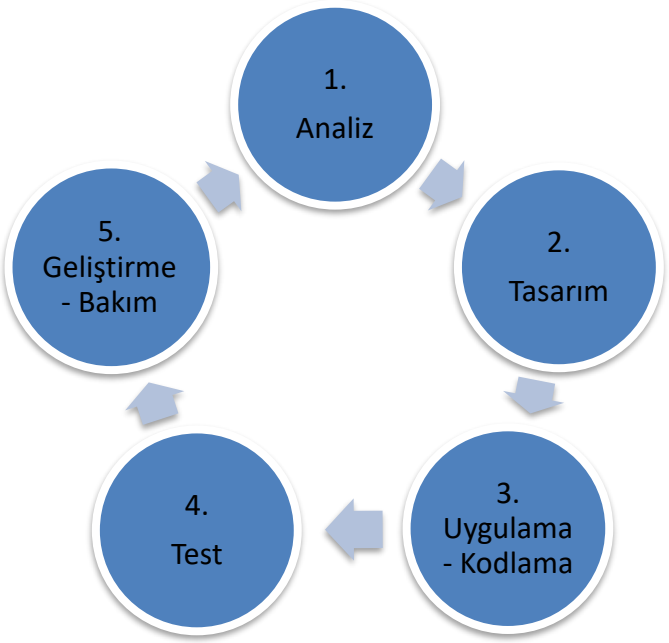
Dokümantasyonu hazırlanır. Analist ve kullanıcılar iş birliği yaparak aşağıdakileri içeren etkili bir dokümantasyon seti oluştururlar:

- Kullanım Prosedürleri ve İş Akış Yönergeleri,
- Çevrimiçi (Online) Yardım Menüleri,
- Sıkça Sorulan Sorular (SSS / FAQ),
- "Beni Oku" ve Kurulum Dosyaları.

Bu dokümanlar, kullanıcılara yazılımın fonksiyonlarını nasıl kullanacaklarını öğretirken, olası hatalarda başvurmaları gereken bir rehber niteliği taşır.
Yazılımın geliştirilmesi aşaması, kendi içinde "Yazılım Geliştirme Yaşam Döngüsü" (Software Development Life Cycle - SLC) adı verilen yinelemeli bir süreci barındırır. Sistem Geliştirme Yaşam Döngüsünden farklı olarak; bu iç döngü sadece yazılımcının *inşa* sürecine odaklanır. Yazılımcı, kendisine gelen teknik tasarım belgesini alır ve Şekil 5'te gösterilen mikro döngüyü işletir:

1. Gelen modülü Analiz eder,
2. Kod yapısını Tasarlar,
3. Kodu yazar (Uygulama),
4. Kendi yazdığı parçayı Test eder (Birim Testi),
5. Gerekirse Geliştirir/Düzenler.

Bu süreç, sistemin bütünü tamamlanana kadar her bir modül için tekrarlanır.



Şekil 5: *Yazılım Geliştirme Aşamasındaki İç Döngü*

Teknik Tasarım Dokümanı'na sadık kalınarak başlatılan geliştirme süreci, genellikle veri katmanının inşası ile başlar. İlişkisel veritabanı (RDBMS) mimarisi tercih edildiyse, tasarım aşamasında çizilen ER diyagramları referans alınarak Oracle, MySQL, PostgreSQL veya MS SQL Server üzerinde fiziksel tablolar oluşturulur. Ancak proje; büyük veri, sosyal medya akışları veya gerçek zamanlı veri analitiği içeriyorsa; Cattell'in (2011) belirttiği NoSQL (Not Only SQL) yaklaşımı benimsenmeli ve bu tip yapılandırılmamış veriler için MongoDB veya Cassandra gibi platformlar tercih edilmelidir.

Örnek uygulamada önce veritabanında "Müşteriler", "Ürünler" ve "Siparişler" tabloları yaratılır ve birbirine bağlanır. Daha sonra yazılım tek bir blok halinde değil, yönetilebilir küçük parçalar halinde yazılır. Uygulamada yazılımcılar müşteri tarafına bakan yüzünü tasarlamak için ön yüz (frontend) ve fonksiyonel süreçleri kodlamak için de arka yüz (backend) kodlamalarını yaparlar. Bir yazılımcı, satış elemanının elindeki tablet

ekranlarını kodlar, tasarımda belirlenen "büyük butonları" ve "kırmızı uyarı renklerini" ekrana yerleştirerek mobil uygulama için ön yüz kodlamayı gerçekleştirir. Başka bir yazılımcı ise uygulamanın çalışmadığı zamanlarda bile veriyi işleyen sunucu kodlarını yazarak arka yüz/plan kodlamayı yapar. Örnek uygulamada yer alan en kritik özellik olan "İnternet yokken çalışma" özelliği burada kodlanır. Arka plan yazılımcı, *"İnternet koptuğunda veriyi tabletin hafızasına kaydet, internet gelince sunucuya yolla"* komutunu yazar. Kodlama aşamasında diğer modül ve yazılımlarla entegrasyon işlemleri de yapılmaktadır. Yazılımcı, mobil uygulamadan "Siparişi Gönder" butonuna basıldığında, bu verinin muhasebe programına (Logo/Netsis vb.) otomatik fatura taslağı olarak düşmesini sağlayan "köprüyü" (API) kodlayarak sistemin doğru işlemesini sağlar.

Bir projenin ömrünü belirleyen kısım olan kullanım kılavuzlarının hazırlanması bu aşamada yapılır, ancak çoğunlukla ihmal edilir. Yazılımı yazan kişi işten ayrıldığında, yerine gelen kişinin sistemi anlayabilmesi için kodların içinde neyin ne işe yaradığını anlatan açıklamalar (yorum satırları), veritabanı şemaları, API bağlantı noktalarının yer aldığı teknik personel için *Sistem Dokümantasyonu* hazırlanması gerekir. Örnek uygulamada *Sipariş modülü, stok kontrolünü X prosedürü ile yapar. Eğer stok 0 ise Y hatasını döndürür"* gibi teknik notlar kaydedilir.

Sistemi kullanacak olan satış personeli ve depo görevlileri için hazırlanan, teknik terimlerden uzak, bol resimli kılavuzlardır Kullanıcı Dokümantasyonu olarak hazırlanmalı. *"Sipariş Nasıl Girilir?", "Yanlış Sipariş Nasıl İptal Edilir?"* gibi başlıklar içeren bir el kitapçığı PDF formatında ve/veya video olarak hazırlanmalı.

Yazılım geliştirme süreci doğrusal değildir; sürekli değişiklik ve geri dönüşler içerir. Loeliger ve McCullough'un (2012) vurguladığı gibi, kaynak kodun tarihçesini tutmak ve ekip çalışmasını senkronize etmek için Git gibi Sürüm Kontrol Sistemleri kullanılmalıdır. Bu sayede, çalışan bir kodun yanlışlıkla bozulması durumunda sistem önceki kararlı sürüme döndürülebilir. Bu adımda kodlama standartlarına uyulmazsa veya dokümantasyon yapılmazsa, sistem çalışsa bile "bakımı yapılamayan", hantal bir yapıya dönüşür. Bir sonraki adım olan **Test** aşamasına, çalışan ve belgelenmiş bir ürün teslim edilmelidir.

6. Sistemin Test Edilmesi ve Sürdürülmesi

Bu aşama, yazılımın canlıya (production) alınmadan önceki son çıkış kapısı ve canlıya alındıktan sonraki yaşam mücadelesidir. Glenford Myers'ın (1979) klasik tanımıyla test; "bir hatayı bulmak amacıyla programı çalıştırma sürecidir". Yazılımcının "Benim bilgisayarımda çalışıyor" demesi, sistemin sahada, gerçek hayatta ve farklı donanımlarda sorunsuz çalışacağının garantisi değildir. Bu evre, sistemin kalite kontrolünden geçtiği ve geleceğe hazırlandığı stratejik bir süreçtir.

Bilgi sistemi son kullanıcıya teslim edilmeden önce, hataların maliyeti düşükken yakalanması hedeflenir. Test süreci, yapay örnek verilerle başlar ve risk azaldıkça gerçek verilerle devam eder. Test türlerinin sınıflandırılması aşağıda gıda toptancısı örneği üzerinden verilmektedir (Pressman, 2014).

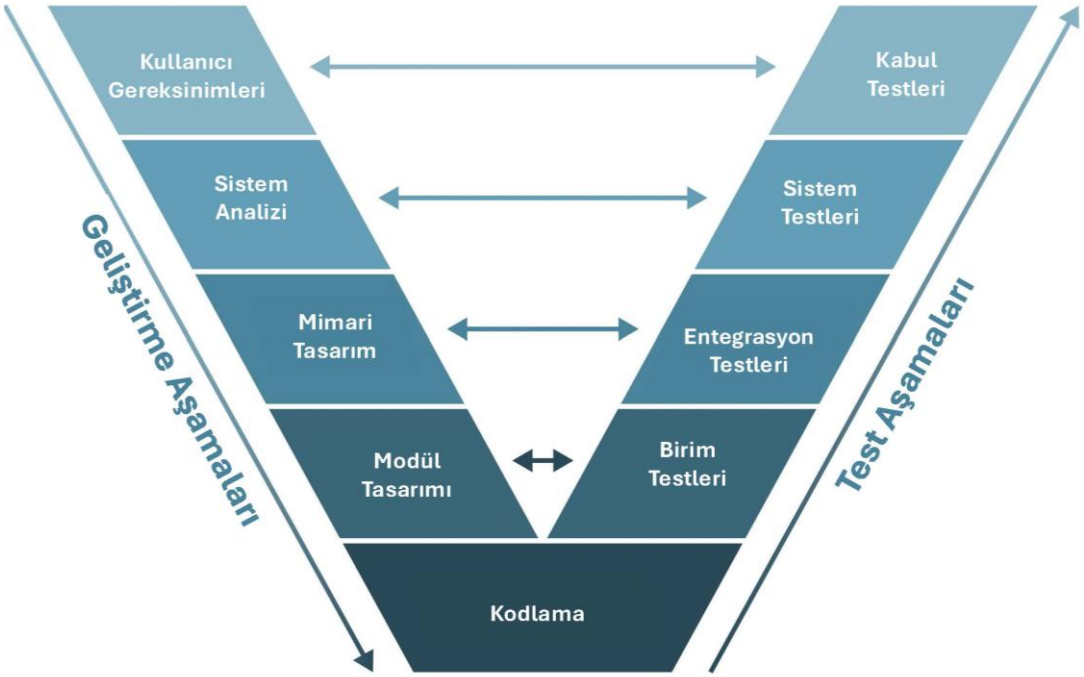
Sistemin test edilmesi ile yazılımda hataların (bug) bulunması amaçlanarak kalite kontrolü yapılır. Gıda toptancı örneğinde yer alan tablet uygulamada girilen bir siparişim muhasebe sisteminde siparişin otomatik olarak oluşup oluşmadığına göre *birim ve entegrasyon testleri* yapılır. Satış temsilcilerinin aynı anda sisteme veri girmesi gibi 1000 kişinin girişi ile sistemin bunlara yanıtının test edildiği *stres ve performans testi* yapılır. İnternetin olmadığı durumda siparişin alınması durumunda sistemin siparişi çevrimdışı iken donanımda tutup sonra bunu merkeze iletip iletmediğinin anlaşıldığı *kurtarma testi* yapılır. En son olarak teknik ekibin geri planda durup gerçek kullanıcıların gerçek ortamlarda, sahada, depoda eldivenle uygulamayı denediği *kullanıcı kabul testi* yapılır. Her türlü olumsuzluk durumunda hatanın mantıksal bir süreçten ortaya çıktığı anlaşılırsa analiz ve tasarım aşamalarına, yazılımdan kaynaklandığı anlaşılırsa da kodlama aşamalarına dönülerek sistem gözden geçirilerek düzeltmeler yapılır. Herhangi bir sebeple son kullanıcının yeni sistemi kullanmaması ile ilgili geri bildirim çok değerlidir ve dikkate alınması gerekir.

Tüm testler bitip sistem canlıya alınıp kullanılmaya başlansa da süreç bitmez. Çünkü yaşayan bir organizma olan yazılım sürekli bakıma ihtiyaç duyar. Sistem canlıda iken gözden kaçan hataların ortaya çıkması ile *düzeltici bakım* yapılır. 3 ay sorunsuz çalışan gıda toptancısı örnek uygulamada bir gün bir müşteri, isminde "Ş, Ğ, Ü" gibi Türkçe karakterler olduğu için faturanın hatalı basıldığını fark eder yazılımcılar bu karakter hatasını düzeltmek için bir yama (patch) yayınlayarak sorunu çözer. Buna benzer olarak yazılımda hata yoktur ama devletin vergi oranlarını değiştirmesi ile sistemdeki formüllerin değişmesi gerektiğinde *uyarlayıcı bakım* yapılır ve yeni sürüm yayınlanır. Sistem kullanıldıkça yeni beklentilerin/taleplerin olması ile *iyileştirici bakım* yapılır. Satış müdürü sistemden memnuniyetini dile getirip satış temsilcilerinin yerinin canlı olarak haritada görünmesinin iyi olacağını belirtmesi ile uygun görüldüğünde projelendirilerek yeni versiyonda yapılması planlanır.

Bu aşama projenin güvenilirliğinin kanıtlandığı, yapılan uygulamanın doğru çalışması ile *çalışma testi*, kullanıcının işine yarıyor ise *kabul testi* ve uzun süre kullanılabilecek durumda ise de *sürdürülebilirliği* test edilmiş olur.

Yapılan sistem iyi test edilmezse, gıda toptancısı örneğinde ilk gün sahada siparişler karışır, veri kayıpları olur ve personel eski kağıt-kalem düzenine dönmek isterse bu projenin ölümü demektir. Bu adım, bu kaosu önleyen sigortadır.

Burada özellikle vurgulamak gerekir ki test işlemi sadece projenin sonunda yapılan bir "hata avcılığı" değildir; projenin en başından (analiz aşamasından) itibaren planlanması gereken bir kalite güvence stratejisidir. Boehm'in (1981) maliyet eğrisine göre; gereksinim aşamasındaki bir hatayı düzeltmek 1 birim maliyetken, canlı aşamada düzeltmek 100 birim maliyettir. Bu kaosu önlemek için V-Modeli (Verification and Validation Model) kullanılır (Rook, 1986).



Şekil 6: Geliştirme ve Test Süreci İçin V Modeli

V-Modeli, geliştirme ve test süreçlerinin birbirinin aynası olduğunu vurgular (Sommerville, 2011). Buna göre modelinin sol tarafı Analiz, Tasarım, Kodlama gibi geliştirme adımlarını temsil ederken sağ tarafı her geliştirme adımının karşısındaki bir test seviyesini temsil etmektedir.

Gıda toptancısı örneğinde; Kullanıcı Gereksinimi aşamasında "Sistem internet yokken de çalışmalı" maddesi yazıldığı an, V diyagramının sağ tarafındaki karşılığı olan Kabul Testi senaryosu da yazılır: "Kullanıcının internetini kes ve sipariş kaydetmeyi dene." Böylece test yapmak için kodun bitmesi beklenmez. Eğer bu model kullanılmazsa, projenin sonunda "Aslında biz böyle bir ekran istememiştik" denildiğinde 1. Adıma dönmek gerekir ki bu, projenin ölümü demektir. V-Modeli, bu riski yöneten sigortadır.

7. Sistemin Gerçekleştirilmesi ve Değerlendirilmesi

Bu aşama, Sistem Geliştirme Yaşam Döngüsü'nün nihai evresi olup, önceki altı adım boyunca sarf edilen entelektüel ve teknik emeğin somut bir iş değerine dönüştüğü yerdir. Literatürde Uygulamaya Geçiş (Implementation) olarak adlandırılan bu safha, O'Brien ve Marakas'ın (2011) belirttiği üzere projenin en yüksek risk taşıyan dönemidir. Zira test ortamında (sandbox) kusursuz çalışan bir sistem, yanlış bir geçiş stratejisi veya yetersiz değişim yönetimi nedeniyle canlı ortamda (production) kurumsal bir kaosa neden olabilir.

Sistemin gerçekleştirilmesi aşamasında kurulum ve dijital sisteme geçiş stratejisi belirlenir. Laudon ve Laudon (2016), dört temel geçiş stratejisi tanımlamaktadır. Belirlenen bir tarihte eski sistemin kapatılıp yine belirlenen bir tarih ve saatte yeni sisteme geçiş stratejisinin seçildiği *doğrudan geçiş* (gıda toptancısı örneğinde Cuma 23:59 da eski sistemin - kağıt ile işlemin - kapatılıp Pazartesi 08:00 da yeni sistemde tablete geçilmesi.

Çok riskli, mobil uygulamada sorun olursa sistem çöker), belli bir süre için aynı anda her iki sistemin de birlikte çalıştığı ve kullanıldığı (en güvenli strateji olup iş yükünü iki katlamakta) *paralel geçiş*, belli bir bölgede yeni sisteme geçişin planlandığı (değişime Ege Bölgesinde başlayıp, orada başarılı olursa diğer bölgelere geçme) *pilot geçiş* ve belli aralıklarla yeni sistem modüllerinin uygulandığı (önce sadece satış modülünü uygulamaya alıp 1 ay sonra stok modülü ve sonra muhasebe modülü ile uygulaya geçiş) *aşamalı geçiş* stratejilerinden biri tercih edilmelidir.

Farklı formatlarda (Excel, kağıt, eski veritabanı) saklanan verilerin yeni tasarlanan ilişkisel veritabanına aktarılması süreci de bu aşamada yer almaktadır. Bu işleme veri göçü denilmektedir. Bu süreçte ETL (Extract, Transform, Load) prosedürleri uygulanır (Inmon, 2005). Özellikle müşteri cari hesapları (borç/alacak) gibi finansal ve hassas verilerin aktarımında oluşabilecek bir veri kirliliği, işletmeyi yasal ve mali zarara uğratabilir. Bu nedenle veri aktarımı, doğrulama algoritmalarıyla defalarca kontrol edilmelidir.

Sistem kurulup canlıya alınıp 3-6 ay geçtikten sonra sistemin değerlendirilmesi yapılmalıdır. Yeni sistemde belirlenen hedeflere ulaşılıp ulaşılmadığı birinci aşamada belirlenmiş olan hedeflerle bugün ulaşılmış olan hedefler karşılaştırılır. Toptan gıda örneğinde belirlenmiş olan hatalı sipariş oranını %15'ten %1'in altına indirmek hedefinde sistem raporlarına bakılır ve iade oranı %2 ye düşmüş ise sistemin hedefe yaklaştığı zamanla daha iyi olacağı öngörülür. Sipariş işleme süresini 45 dakikadan 2 dakikaya düşürmek hedefinde ise siparişlerin 3 saniyede muhasebeye düştüğü tespit edilerek başarı teyit edilir. Diğer hedef olan stok doğruluğu depo sayımı ile sistemdeki rakamın uyuştuğu tespit edilerek olmayan ürünü satma durumu ortadan kalmış oluyor.

Yeni sistemin bakım planını devreye alınması ile proje ekibinin görevi sona erer. Ürün şirketin Bilgi İşlem departmanı sorumluluğunda sunucuların, veritabanı sisteminin ve eldeki donanım ürünlerinin süreç içerisinde bakım ve yenileme çalışmaları/sözleşmeleri yapılır. Sistemin çökmesi ile kimlerin ayağa kaldıracağı, destek hattının nasıl çalışacağı belirlenmelidir.

Sistem ne kadar kullanıcı dostu tasarlanırsa tasarlansın, teknoloji kabulü için Kullanıcı Eğitimi şarttır. Kotter'ın (1996) değişim yönetimi ilkelerine göre; kullanıcıların direnç göstermemesi için sistemin faydaları anlatılmalı ve kullanıcıların sistemi eksiksiz kullanabildiği sertifikasyon süreçleriyle teyit edilmelidir.

Sistem geliştirmenin son safhasında analist bilgi sisteminin gerçekleştirilmesini sağlar. Eski sistemden yeni sisteme geçiş için bir plan oluşturur. Bu süreç dosyaları eski formattan yeni formata dönüştürmeyi yada bir veri tabanı oluşturmayı kapsar.

Değerlendirme aşaması, projenin bittiği değil, sistemin yaşam döngüsünün başladığı yerdir. Dennis ve Wixom'un (2015) vurguladığı gibi SGYD döngüseldir. Değerlendirme raporunda tespit edilen yeni bir darboğaz veya eksiklik, analisti tekrar bir önceki safhaya veya döngünün başına dönmeye zorlayabilir. Bu, yaşayan bir organizma olan bilgi sisteminin evrimsel sürecidir.

5 | SONUÇ

Günümüzün rekabetçi ve karmaşık iş dünyasında, bilişim sistemleri yalnızca operasyonel birer araç değil, stratejik dönüşümün temel taşıyıcılarıdır. Bu çalışma, karmaşık problemlerin çözümünde bütüncül bir bakış açısı sunan Sistem Yaklaşımı ile disiplinli bir uygulama süreci sağlayan Sistem Geliştirme Yaşam Döngüsü (SGYD) metodolojisinin entegrasyonunu ele almıştır.

Makalede yapılan analizler, SGYD'nin işletme problemlerinin çözümünde stratejik bir araç olduğunu ortaya koymuştur. SGYD; planlama, analiz, tasarım, geliştirme, test etme, uygulama ve bakım aşamalarıyla, soyut iş problemlerini somut, sürdürülebilir ve ölçülebilir çözümlere dönüştüren hayati bir kılavuzdur. Bu sistematik yaklaşım, işletmelerin süreçlerini optimize ederken yenilikçi çözümler geliştirmesine olanak tanımakta ve farklı sektörlerin özgün ihtiyaçlarına göre uyarlanabilen esnek bir yapı sunmaktadır.

Özellikle çalışmada sunulan **Tablo 1**, Kendall ve Kendall'ın (2019) metodolojik yaklaşımı çerçevesinde; SGYD'nün soyut bir teoriden somut bir yönetim planına nasıl dönüştüğünü gösteren stratejik bir haritadır. Bir bilişim projesinin fikir aşamasından (1. Adım) başlayıp canlı kullanıma (7. Adım) geçene kadar geçirdiği evrimi; sorumlular, eylemler ve üretilen belgeler (çıktılar) ekseninde özetleyen bu matris, yazılım geliştirme sürecinin sadece teknik bir 'kodlama' işi değil; insan, süreç ve teknolojinin koordinasyonunu gerektiren disiplinli bir yönetim süreci olduğunu kanıtlamaktadır.

Tablo 1: Sistem Geliştirme Yaşam Döngüsü (SGYD) Süreç, Faaliyet ve Çıktı Matrisi

Aşamalar	Sorumlu Kişi	Katılımcılar	Amaç	Faaliyetler	Çıktıları
1. Problemlerin, Fırsatların ve Amaçların Tanımlanması	Proje Sorumlusu	• Kullanıcılar • Analistler • Projeyi düzenleyen yöneticiler	Problemler, fırsatlar ve amaçların netleştirilmesi	• Müşteri/Yönetim görüşmeleri • Mevcut durum analizi • Bilginin özetlenmesi • Proje sınırlarının tahminlenmesi	• Fizibilite raporu • Projenin amaç raporu
2. Bilgi Gereksinimlerinin Belirlenmesi	Sistem Analisti	• Analistler • Kullanıcılar • Operasyon yöneticileri	Hangi bilgiye, nerede ve ne zaman ihtiyaç duyulduğunun araştırılması	• Mülakatlar ve anketler • Verinin örneklenmesi • Karar vericinin davranışları • İş akış gözlemi • Prototip oluşturulması	• Bilgi gereksinim listesi • Veri akış şemaları
3. Sistem İhtiyaçlarının Analizi	Sistem Analisti	• Yazılım Mimarları • Departman Yöneticileri	Doğru kaynağın doğru zamanda kullanılabilmesi için mantıksal yapının kurulması.	• Piyasa ve maliyet analizi • Personel ve zaman planlaması • Karar tablolarının hazırlanması	• Taslak proje planı • Sistem Önerisi Raporu
4. Önerilen Sistemin Tasarımı	Proje Sorumlusu, Sistem Analisti	• Kullanıcılar (Arayüz onayı için) • Yazılımcılar • Veritabanı Uzmanları	Sistemin "Nasıl" çalışacağının (fiziksel ve mantıksal) belirlenmesi.	• Ekran tasarımları • Veritabanı şemalarının çizimi • Ağ ve güvenlik mimarisi • Donanım seçimi	• Sistem Tasarım Dokümanı • Veritabanı Şemaları
5. Yazılımın Geliştirilmesi ve Belge Yönetimi	Yazılımcı	• Sistem Analisti • Teknik Yazarlar • Test Uzmanları	Tasarımın çalışan bir ürüne dönüştürülmesi ve kayıt altına alınması.	• Kodlama/ Programlama • Kullanıcı kılavuzlarının yazılması • Sistem dokümantasyonu	• Yazılım
6. Sistemin Test Edilmesi ve Sürdürülmesi	Proje Sorumlusu	• Analistler • Yazılımcılar • Test ekibi • Kullanıcılar	Gerçek uygulamada çıkabilecek sorunları tespit etme ve giderme	• Test planının oluşturulması • Birim, Entegrasyon ve Stres testleri • Kullanıcı Kabul Testleri	• Hata Raporları • Test Sonuç Belgeleri • Onaylanmış Yazılım
7. Sistemin Gerçekleştirilmesi ve Değerlendirilmesi	Proje Sorumlusu	• Gerçek kullanıcılar • Sistem Destek Ekibi	Sistemin canlıya alınması, verilerin taşınması ve başarının ölçülmesi	• Kurulum ve veri aktarımı • Kullanıcı eğitimleri • Sistemin performans izlemesi • Hedef-Sonuç karşılaştırması	• Canlı Sistem • Değerlendirme Raporu • Saha Raporları

Ayrıca çalışmanın kavramsal çerçevesinde, başarılı bir bilişim sistemi projesinin; donanım, yazılım veya veri bileşenlerinin tek başına mükemmelliği ile değil, tüm bu bileşenlerin net bir "Amaç ve Hedef" doğrultusunda birleşmesiyle mümkün olduğu vurgulanmıştır. Buna ek olarak, geleneksel Yönetim Bilişim Sistemleri piramidinin, İş Zekası (Geçmiş Odaklı) ve İş Analitiği (Gelecek Odaklı) yetkinlikleri ile genişletilmesi gerektiği

ortaya konmuştur. Bu yeni model, yöneticilere sadece "ne olduğunu" değil, "ne olacağını" da göstererek proaktif karar alma imkânı sağlamaktadır.

Bu çalışma, problem çözme metodolojileri üzerine teorik ve kavramsal bir çerçeve sunmaktadır. Çalışmanın temel kısıtı (sınırlılığı), önerilen "Analitik Yetkinlik Tabanlı Genişletilmiş YBS Modeli"nin belirli bir sektörde veya işletmede vaka analizi yöntemiyle test edilmemiş olmasıdır. Modelin geçerliliği, tecrübelere ve literatürdeki teorik temellere dayanmaktadır.

Gelecek çalışmalarda, bu makalede önerilen modelin bir üretim veya hizmet işletmesindeki dijital dönüşüm projesinde uygulanması ve elde edilecek ampirik (sayısal) sonuçlarla SGYD'nin verimliliğe etkisinin ölçülmesi önerilmektedir. Ayrıca uygulayıcılar (sektör profesyonelleri) için; projelerine başlarken sadece teknik fizibiliteye değil, çalışmada vurgulanan "Amaç ve Hedefler" bileşeni ile "Veri Analitiği" yetkinliklerine öncelik vermeleri, proje başarısızlık oranlarını düşürecek somut bir adımdır.

Sonuç olarak; hiçbir metodoloji yetkin bir analistin yerini tutamaz, ancak SGYD gibi güçlü metodolojiler yetkin analistlerin elinde işletmelere rekabet avantajı sağlayan en güçlü silaha dönüşür. Geleceğin başarılı organizasyonları, teknolojiyi sadece satın alanlar değil; onu bu sistematik yaklaşımlarla iş süreçlerine en iyi entegre edenler olacaktır.

KAYNAKÇA

Ackoff, R. L. (1981). *Creating the corporate future: Plan or be planned for*. New York, NY: John Wiley & Sons.

Anthony, R. N. (1965). *Planning and control systems: A framework for analysis.* Boston, MA: Harvard University.

Avison, D., & Fitzgerald, G. (2006). *Information systems development: Methodologies, techniques and tools* (4th ed.). Maidenhead, UK: McGraw-Hill.

Boehm, B. W. (1981). *Software engineering economics.* Englewood Cliffs, NJ: Prentice-Hall.

Boulding, K. E. (1956). General systems theory—the skeleton of science. *Management Science, 2*(3), 197-208.

Cattell, R. (2011). Scalable SQL and NoSQL data stores. *SIGMOD Record, 39*(4), 12-27.

Checkland, P. (1981). *Systems thinking, systems practice.* Chichester, UK: John Wiley & Sons.

Checkland, P., & Scholes, J. (1990). Soft systems methodology in action. Chichester, UK: John Wiley & Sons.

Chen, P. P. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS), 1*(1), 9-36.

Davenport, T. H. (2006). Competing on analytics. *Harvard Business Review, 84*(1), 98-107.

Davis, G. B. (1982). Strategies for information requirements determination. *IBM Systems Journal, 21*(1), 4-30.

Davis, W. S. (1999). *Systems analysis and design: A structured approach.* Reading, MA: Addison-Wesley.

DeLone, W. H., & McLean, E. R. (2003). The DeLone and McLean model of information systems success: A ten-year update. *Journal of Management Information Systems, 19*(4), 9-30.

Dennis, A., Wixom, B. H., & Roth, R. M. (2015). *Systems analysis and design* (6th ed.). Hoboken, NJ: Wiley.

Gane, C., & Sarson, T. (1979). *Structured systems analysis: Tools and techniques.* Englewood Cliffs, NJ: Prentice-Hall.

Hoffer, J. A., George, J. F., & Valacich, J. S. (2017). *Modern systems analysis and design* (8th ed.). Boston, MA: Pearson.

Hofstede, G. (2001). *Culture's consequences: Comparing values, behaviors, institutions and organizations across nations* (2nd ed.). Thousand Oaks, CA: Sage Publications.

IEEE. (2008). *IEEE standard for software and system test documentation (IEEE Std 829-2008).* New York, NY: IEEE.

Inmon, W. H. (2005). *Building the data warehouse* (4th ed.). Indianapolis, IN: Wiley.

Kahneman, D. (2011). *Thinking, fast and slow.* New York, NY: Farrar, Straus and Giroux.

Kaplan, A. (1964). *The conduct of inquiry: Methodology for behavioral science.* San Francisco, CA: Chandler Publishing.

Kendall, K. E., & Kendall, J. E. (2019). *Systems analysis and design* (10th ed.). New York, NY: Pearson.

Kotter, J. P. (1996). *Leading change.* Boston, MA: Harvard Business School Press.

Laudon, K. C., & Laudon, J. P. (2016). *Management information systems: Managing the digital firm* (14th ed.). New York, NY: Pearson.

Laudon, K. C., & Laudon, J. P. (2020). *Management information systems: Managing the digital firm* (16th ed.). Hoboken, NJ: Pearson.

Loeliger, J., & McCullough, M. (2012). *Version control with Git: Powerful tools and techniques for collaborative software development* (2nd ed.). Sebastopol, CA: O'Reilly Media.

Martin, J. (1991). *Rapid application development.* New York, NY: Macmillan Publishing Co.

Myers, G. J. (1979). *The art of software testing.* New York, NY: John Wiley & Sons.

Norman, D. A. (2013). *The design of everyday things: Revised and expanded edition.* New York, NY: Basic Books.

O'Brien, J. A., & Marakas, G. M. (2011). *Management information systems* (10th ed.). New York, NY: McGraw-Hill/Irwin.

Open University. (2000). *Understanding systems: The system as a whole.* Milton Keynes, UK: Open University.

Pressman, R. S. (2014). *Software engineering: A practitioner's approach* (8th ed.). New York, NY: McGraw-Hill Education.

Rook, P. (1986). Controlling software projects. *Software Engineering Journal, 1*(1), 7-16.

Rüping, A. (2003). *Agile documentation: A pattern guide to producing lightweight documents for software projects.* Chichester, UK: John Wiley & Sons.

Schwab, K. (2016). *The fourth industrial revolution.* New York, NY: Crown Business.

Senge, P. M. (1990). *The fifth discipline: The art & practice of the learning organization.* New York, NY: Doubleday/Currency.

Shneiderman, B., Plaisant, C., Cohen, M., & Jacobs, S. (2016). *Designing the user interface: Strategies for effective human-computer interaction* (6th ed.). Boston, MA: Pearson.

Simon, H. A. (1960). *The new science of management decision.* New York, NY: Harper & Row.

Simon, H. A. (1977). *The new science of management decision* (Rev. ed.). Englewood Cliffs, NJ: Prentice-Hall.

Sommerville, I. (2011). *Software engineering* (9th ed.). Boston, MA: Addison-Wesley.

Standish Group. (2015). *The chaos report.* Boston, MA: The Standish Group International.

Swanson, E. B. (1976). The dimensions of maintenance. In *Proceedings of the 2nd International Conference on Software Engineering* (pp. 492–497). Los Alamitos, CA: IEEE Computer Society.

Tecim, V. (2025a). *Sistem geliştirme yaşam döngüsü: Temel kavramlar.* Erişim tarihi: 10 Eylül 2025, https://vahaptecim.com.tr/?page_id=167

Tecim, V. (2025b). *Sistem geliştirme yaşam döngüsü: Uygulama adımları.* Erişim tarihi: 10 Eylül 2025, https://vahaptecim.com.tr/?p=366

Valacich, J. S., & George, J. F. (2020). *Modern systems analysis and design* (9th ed.). Boston, MA: Pearson.

von Bertalanffy, L. (1968). *General system theory: Foundations, development, applications.* New York, NY: George Braziller.

Wetherbe, J. C. (1984). Systems analysis and design: Traditional, structured, and advanced concepts (2nd ed.). St. Paul, MN: West Publishing.

Whitten, J. L., & Bentley, L. D. (2007). *Systems analysis and design methods* (7th ed.). New York, NY: McGraw-Hill/Irwin.

Wiegers, K., & Beatty, J. (2013). *Software requirements* (3rd ed.). Redmond, WA: Microsoft Press.

Wilson, B. (1990). *Systems: Concepts, methodologies, and applications* (2nd ed.). Chichester, UK: John Wiley & Sons.

Wolstenholme, E. F. (1990). *System enquiry: A system dynamics approach.* Chichester, UK: John Wiley & Sons.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal, 26*(3), 276-292.