T.C. SANAYİ VE TEKNOLOJİ BAKANLIĞI

#MİLLİ TEKNOLOJİ HAMLESİ

TÜBİTAK BİLGEM

YEŞİL YAZILIM MÜHENDİSLİĞİ

ARAŞTIRMA SERİSİ - SAYI 36

TÜBİTAK BİLGEM

YAZILIM TEKNOLOJİLERİ ARAŞTIRMA ENSTİTÜSÜ

Yeşil Yazılım Mühendisliği

ARAŞTIRMA SERİSİ - SAYI 36

Yazar

Gökçenur ÇINAR

Yayın Koordinatörü

Kübra ERTÜRK

Editörler

Ezgi TAŞKOMAZ

Tuğçe YILMAZ

Sinem ALTUĞ GÖL

Tasarım

Özge DOĞAN

İletişim: 0(312) 289 92 22 - yte.bilgi@tubitak.gov.tr

https://bilgem.tubitak.gov.tr/yte/

Ön Söz

TÜBİTAK BİLGEM Yazılım Teknolojileri Araştırma Enstitüsü (YTE), 2012 yılından bu yana yazılım teknolojilerinde Ar-Ge faaliyetleri yürüten bir araştırma kuruluşudur. Araştırma faaliyetlerinde elde ettiği birikimini stratejik, hassas ve kritik projeler yürüterek kamu adına hayata geçirmekte; kurumlarımıza dijital dönüşüm, yazılım geliştirme teknolojileri ve kalite süreçleri konusunda danışmanlık vermektedir.

TÜBİTAK BİLGEM YTE tarafından hazırlanan Araştırma Serisi ile kurum içi içerik üretme çalışmalarının yaygınlaştırılması ve hazırlanan içeriklerin sektörün erişimine açılması amaçlanmaktadır. Araştırma Serisi'nde yayınlanan çalışmalar TÜBİTAK BİLGEM YTE çalışanlarının projelerde elde ettiği bilgi birikimini paylaşmak adına derlenmiştir. Bu çalışmalar ile ülkemizin yazılım sektörüne katkı sağlanılması hedeflenmektedir.

İçindekiler

Giriş

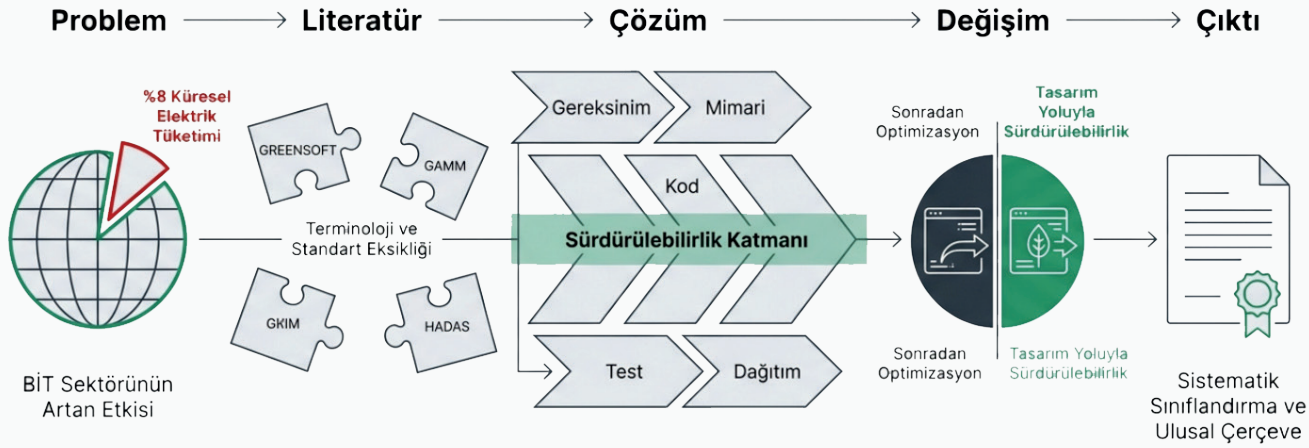
Yazılım sistemlerinin küresel ölçekte enerji tüketimi ve karbon salınımı üzerindeki etkisi giderek artmaktadır. Bilgi ve iletişim teknolojileri [Information and Communication Technology (ICT)] sektörü, dünya genelinde elektrik tüketiminin yaklaşık yüzde sekizini oluşturmakta ve bu oran her geçen yıl büyümektedir. Donanım bileşenlerinin enerji verimliliğine yönelik çalışmalar hız kazanmış olsa da, yazılım bileşenlerinin çevresel etkileri henüz aynı düzeyde incelenmemiştir. Yazılımların mimarisi, algoritma seçimi, veri yönetimi ve çalışma süresi gibi faktörler enerji tüketimini doğrudan etkilemektedir.

Bu araştırma serisinin amacı, Yeşil Yazılım Mühendisliği (YYM) kavramını kuramsal açıdan inceleyerek yazılım mühendisliği süreçlerinde çevresel sürdürülebilirliğin nasıl sağlanabileceğini ortaya koymaktır. Seri, mevcut literatürde yer alan tanım, yöntem, metrik ve çerçeveleri sistematik biçimde analiz etmeyi ve bu alandaki eksiklikleri belirlemeyi hedeflemektedir.

Yazılım mühendisliği süreçlerinde enerji tüketimi, kaynak kullanımı ve karbon ayak izi ölçümü üzerine yapılan çalışmalar son yıllarda artmış olsa da, Yeşil Yazılım Mühendisliği (YYM) alanında henüz ortak bir tanım, ölçüm yöntemi veya endüstri standardı bulunmamaktadır. Bu durum, araştırmaların birbirinden kopuk yürütülmesine ve elde edilen bulguların karşılaştırılabilirliğinin düşmesine neden olmaktadır.

YYM alanında yapılan çalışmalarda "Green IT (Yeşil BT)", "Sustainable Software Engineering (Sürdürülebilir Yazılım Mühendisliği)", "Energy-aware Software (Enerji Farkındalığına Sahip Yazılım)" gibi terimler birbiriyle karıştırılmakta; "Green in" ve "Green by" ayrımı çoğu zaman net biçimde tanımlanamamaktadır. Bunun sonucu olarak, geliştirilen modeller farklı disiplinlerde benzer sorunları yeniden ele almakta ve alanın bilimsel olgunlaşma süreci yavaşlamaktadır. Şekil 1' de yazılımın çevresel ayak izi bu kavramlar doğrultusunda gösterilmiştir.

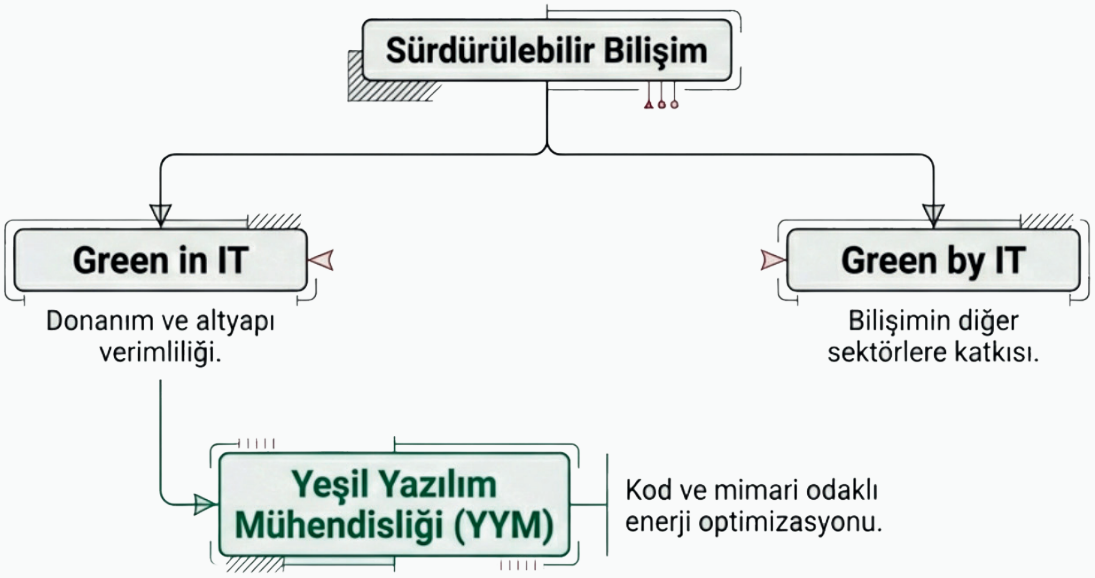
Yazılım mühendisliği alanında çevresel sürdürülebilirliği destekleyen ortak bir kuramsal çerçeve, ölçüm standardı ve metodoloji eksikliği bulunmaktadır.



Şekil 1. Yazılımın Çevresel Ayak İzi

Yeşil Bilgi Teknolojileri Kavramının Gelişimi

Bilgi teknolojilerinin ölçeğinin büyümesi, enerji tüketimi ve emisyon etkisinin daha görünür hâle gelmesine yol açmış bu da "Green IT" yaklaşımını BT alanında önemli bir araştırma ve uygulama gündemi haline gelmesine neden olmuştur. Green IT, donanım üretiminden sistemlerin işletimine kadar yaşam döngüsü boyunca enerji verimliliği ve çevresel etkinin azaltılmasını hedefleyen bütüncül bir yaklaşım olarak ele alınmaktadır. Şekil 2' de görüldüğü gibi bu yaklaşım literatürde sıklıkla Green in IT ve Green by IT olarak ele alınmaktadır. Green in IT, donanım ve yazılım bileşenlerinin enerji kullanımını azaltmayı hedefleyen uygulamalardır. Green by IT ise BT'nin araç olarak kullanıldığı, sürdürülebilirliğe katkı sağlayan çözümlerdir [1].



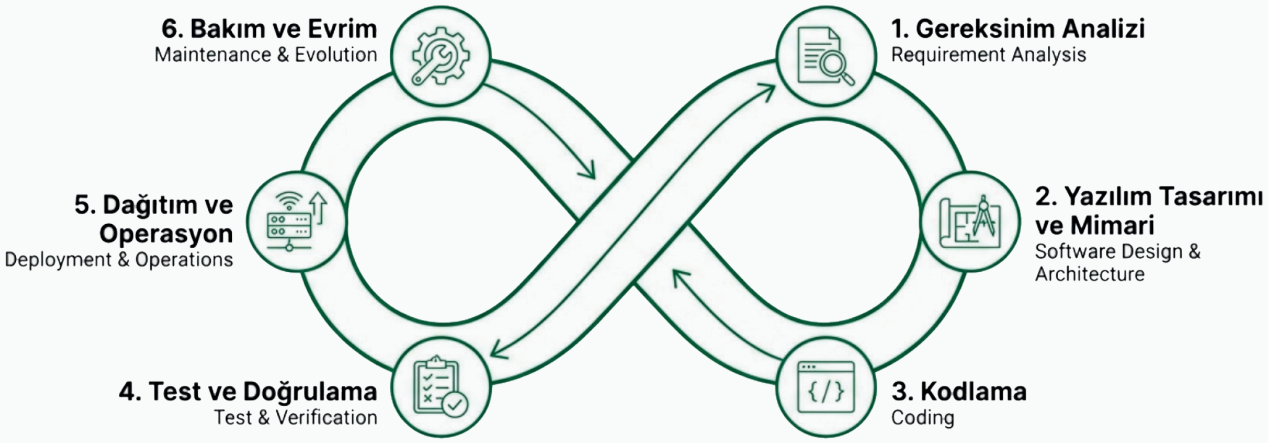
Şekil 2. Kavramsal Çerçeve

Erken dönem çalışmalarında ağırlıklı olarak veri merkezleri ve donanım verimliliğine odaklanıldığı, yazılımın çevresel etkisinin daha sınırlı tartışıldığı görülmektedir [2]. Ancak yazılımın kaynak tüketimini doğrudan şekillendiren bir kontrol katmanı olması, performans, veri hareketi, kaynak yönetimi ve kullanım senaryolarının yazılım kararlarıyla belirlendiği gerçeğini öne çıkarmıştır. Bu nedenle sürdürülebilirlik odağı zamanla yazılımın geliştirilmesi ve işletimine doğru genişlemiş olup "Yeşil Yazılım Mühendisliği" (YYM) kavramını belirginleştirmiştir.

Bu dönüşüm, sürdürülebilirliği altyapı seçimiyle sınırlamamak gerektiğini gösterir. Kurumsal ölçekte etki üretmek için sürdürülebilirlik hedeflerini yazılım mühendisliği süreçlerine (gereksinim, mimari, geliştirme, test, devreye alma ve işletim) de dahil etmek gerekmektedir.

Yeşil Yazılım Mühendisliği'nin Ortaya Çıkışı

Yeşil Yazılım Mühendisliği, yazılım sistemlerinin geliştirilmesi ve işletiminde enerji tüketimini azaltmayı, kaynak kullanımını iyileştirmeyi ve çevresel etkiyi düşürmeyi hedefleyen bir mühendislik yaklaşımıdır ([1], [3]). Bu yaklaşım, sadece kod optimizasyonu değildir. Şekil 3' de gösterilmiş olan yazılım mühendisliği yaşam döngüsü boyunca alınan, gereksinim mühendisliği, mimari tasarım, geliştirme, test, dağıtım ve operasyon kararların tamamı bu hedefi etkilemektedir.



Şekil 3. Sürdürülebilir Yaşam Döngüsü

YYM'nin temel hedefi, yazılım yaşam döngüsündeki her aşamada çevresel farkındalığı artırmaktır. Bu bağlamda, gereksinim mühendisliğinden kodlama, test ve dağıtıma kadar uzanan süreçlerde enerji verimliliği ölçülebilir bir kriter hâline getirilmektedir.

Nurmivaara [3], YYM alanındaki araştırmaları sistematik biçimde incelemiş ve mevcut literatürde ortak terminoloji eksikliğinin en belirgin sorun olduğunu tespit etmiştir. Çalışmada, yazılımın çevresel sürdürülebilirliğini artırmaya yönelik yaklaşımların genellikle iki grupta toplandığı belirtilmiştir:

- **Süreç ve çerçeve öneren yaklaşımlar:** Mimariler, olgunluk modelleri, süreç rehberleri.
- **Özelleştirilmiş ve deneysel yaklaşımlar:** IoT [Internet of Things (Nesnelerin İnterneti)], mobil uygulamalar, çok çekirdekli işlemciler gibi dar alanlarda enerji optimizasyonuna odaklanan çözümler.

Bu ikili yapı, alanın bilimsel olgunluk düzeyini hâlen "erken gelişim" aşamasında tutmaktadır.

Yeşil Yazılım Mühendisliği'nin Boyut ve İlkeleri

Calero ve Piattini [1] sürdürülebilirliği beş boyutta ele alır: bireysel, sosyal, ekonomik, teknik ve çevresel. YYM tartışmaları çoğunlukla çevresel boyuta odaklanır. Ancak kurumsal uygulamada teknik ve ekonomik etkiler (performans, maliyet, güvenilirlik) birlikte yönetilmek zorundadır.

YYM literatüründe pratikte tekrar eden temel ilkeler şunlardır ([1], [4]) :

- **Enerji farkındalığı:** Yazılım bileşenlerinin enerji tüketiminin ölçülmesi ve minimize edilmesi,
- **Kaynak verimliliği:** Bellek, ağ ve depolama kullanımının optimize edilmesi,
- **Yaşam döngüsü perspektifi:** Geliştirme, dağıtım ve bakım aşamalarında çevresel etki değerlendirmesi,
- **Farkındalık ve eğitim:** Yazılım geliştiricilerin sürdürülebilirlik bilinci kazanması,
- **Metrik tabanlı karar verme:** Karbon ayak izi, işlem süresi, enerji/performans oranı gibi ölçütlerle izlenebilirlik.

Bu ilkeler, günümüzde sürdürülebilir yazılım politikalarının ve kurum içi yeşil stratejilerin temelini oluşturmaktadır.

Literatürdeki Yaklaşımlar ve Modeller

Bu bölümde, yeşil yazılım mühendisliği kapsamında literatürde önerilen temel model ve yaklaşımlar özetlenmektedir.

GREENSOFT Modeli

GREENSOFT, yazılım ürünlerinin çevresel etkilerini beş aşamalı yaşam döngüsü üzerinden değerlendiren ilk kapsamlı çerçevedir. Bu döngü geliştirme, dağıtım, kullanım, bakım ve bertaraf adımlarından oluşmaktadır. Model, hem süreç hem ürün odaklı göstergeleri birleştirir ve yazılımın "doğumdan ölüme" kadar olan çevresel ayak izini izlemeyi amaçlar [5].

GAMM [Green-Agile Maturity Model (Sürdürülebilir Çevik Olgunluk Modeli)]

GAMM, sürdürülebilirliği çevik geliştirme pratiklerine entegre etmeyi hedefleyen bir olgunluk yaklaşımıdır [6]. GAMM, Scrum süreçlerine "Yeşil Sprint" kavramını dahil eder ve ekiplerin çevresel etkilerini periyodik olarak ölçmelerine imkân tanır.

HADAS Yaklaşımı

HADAS, enerji tüketimi yüksek bileşenleri tespit edip çalışma zamanında alternatif konfigürasyonlara geçebilen öz-uyarlamalı bir yaklaşım sunar [7]. Bu yaklaşım, özellikle dinamik çalışma ortamlarında sistemin bağlamsal koşullara göre kendini optimize etmesine olanak tanır. Böylece performans–enerji dengesi korunurken, gereksiz kaynak kullanımının önüne geçilmesi hedeflenir. Ayrıca HADAS, karar mekanizmalarını çalışma zamanına taşıyarak statik optimizasyonların ötesinde esneklik sağlar.

GSMRM [Green Software Multi-Sourcing Readiness Model (Yeşil Yazılım Çoklu Tedarik Hazırlık Modeli)]

GSMRM, çok kaynaklı geliştirme ortamlarında yeşil pratiklerin uygulanabilirliğini değerlendiren; tedarik zinciri ve yönetişim boyutuna vurgu yapan bir hazırlık modelidir [8]. Model, farklı paydaşların sürdürülebilirlik hedeflerine uyum düzeyini analiz ederek organizasyonel olgunluğu ölçmeyi amaçlar. Bu yönüyle yalnızca teknik değil, aynı zamanda süreç ve yönetimsel faktörleri de kapsayan bütüncül bir değerlendirme sunar. Ayrıca GSMRM, kurumların sürdürülebilir yazılım stratejilerini çok paydaşlı yapılarda sistematik biçimde planlamasına destek olur.

Ölçüm Metrikleri ve Değerlendirme Yöntemi

YYM'de kullanılan metrikler genel olarak enerji, karbon salınımı, performans, ve verimlilik odaklıdır. En sık karşılaşılan ölçüm yaklaşımları:

- **Enerji tüketimi (Joule veya Watt-saniye):** Yazılım bileşenlerinin çalışma sırasında kullandığı enerji miktarı.
- **Karbon ayak izi (CO_2 eq):** Yazılım sisteminin doğrudan veya dolaylı neden olduğu emisyon miktarı.
- **Enerji-verimlilik oranı (EER):** Enerji tüketimi başına elde edilen işlem çıktısı.
- **Kaynak kullanım yoğunluğu:** Bellek, CPU, I/O gibi kaynakların sürdürülebilirlik açısından izlenmesi.

Son yıllarda, bu metriklerin standartlaştırılması amacıyla çeşitli çerçeveler geliştirilmiştir. Örneğin AWS [9] "Carbon Footprint Tool (Karbon Ayak İzi Aracı)" adlı hizmetiyle bulut kaynaklarının emisyon takibini sağlamaktadır. Literatürdeki güncel stratejik eğilimler [10] ve yapay zekânın 2030 yılına kadar yazılım

mühendisliği üzerindeki etkileri [11], sürdürülebilirliğin terminoloji uyumsuzluğu ve standardizasyon eksikliği gibi sorunlara rağmen temel bir odak noktası haline geleceğini göstermektedir ([3], [12]).

Literatürde Tespit Edilen Eksiklikler

Yapılan kapsamlı incelemeler, YYM alanında aşağıdaki temel eksikliklerin sürdüğünü göstermektedir [3]:

- **Terminoloji uyumsuzluğu:** Green IT, Green Computing ve YYM kavramları arasında sınırlar net değildir.
- **Standardizasyon eksikliği:** Uluslararası düzeyde kabul görmüş ölçüm veya değerlendirme standardı bulunmamaktadır. Elde edilen sonuçların birbirleriyle doğrudan karşılaştırılmasını ve çalışmaların tekrarlanabilirliğini zorlaştırmaktadır.
- **Endüstriyel Katılım Eksikliği:** Akademik modellerin büyük bölümü laboratuvar ölçeklidir; gerçek üretim ortamlarında doğrulama yapılmamıştır.
- **Eğitim ve farkındalık sorunu:** Yazılım mühendisleri enerji tüketimi ve sürdürülebilirlik ölçütlerini çoğunlukla bilmemektedir.
- **Veri erişimi kısıtlılığı:** Enerji tüketimi verileri genellikle tescilli platformlarda saklandığından, araştırmacılar arasında karşılaştırma güçleşmektedir.

Bu eksiklikler, gelecekte YYM alanında ortak bir ölçüm ekosistemi kurulmasının önemini açıkça ortaya koymaktadır.

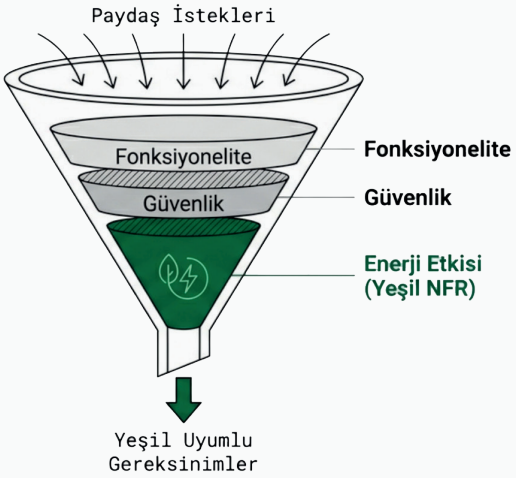
Yöntem

Bu bölümde Yeşil Yazılım Mühendisliği (YYM) yaklaşımının yazılım mühendisliği süreçlerine nasıl entegre edilebileceği açıklanmaktadır. YYM'nin temel hedefi; sürdürülebilirliği "sonradan yapılan optimizasyon" olarak değil, gereksinimden operasyona kadar uzanan uçtan uca bir mühendislik çıktısı olarak yönetmektir. Endüstri odaklı çalışmalar, sürdürülebilirliğin güvenlik ve güvenilirlik gibi temel kalite nitelikleriyle aynı düzeyde ele alınması gerektiğini vurgular ve bunun "sustainability-by-design" yaklaşımıyla mümkün olacağını belirtir ([11], [13]). Akademik literatürde ise YYM'nin ölçüm zorlukları, standart eksikliği ve süreçlere sistematik entegrasyon ihtiyacı öne çıkmaktadır ([3], [14]). Bu nedenle yöntem yaklaşımı, her süreç adımında hangi kararların çevresel etkiyi belirlediğini, hangi kontrol noktalarıyla yönetileceğini ve hangi metriklerle doğrulanacağını netleştirmeyi hedefler.

Gereksinim Analizi

Yazılım geliştirme sürecinin ilk ve önemli aşamalarından biri olan gereksinim analizi, sistemin fonksiyonel ve fonksiyonel olmayan özelliklerinin belirlendiği aşamadır. Bu aşamada alınan kararlar, yazılımın mimarisini, çalışma biçimini ve dolayısıyla yaşam döngüsü boyunca tüketeceği enerji miktarını doğrudan etkilemektedir. Şekil 4'te de gösterildiği gibi Yeşil Yazılım Mühendisliği (YYM) yaklaşımına göre, çevresel sürdürülebilirliğin yazılım süreçlerine entegrasyonu, en etkili biçimde bu erken aşamada gerçekleştirilebilmektedir.

Calero ve Piattini [1], yazılım sistemlerinin çevresel etkilerinin büyük ölçüde gereksinim aşamasında şekillendiğini ve sonraki aşamalarda bu etkinin düzeltilmesinin hem maliyetli hem de sınırlı olduğunu vurgulamaktadır. Bu nedenle YYM, çevresel sürdürülebilirliği sonradan eklenen bir iyileştirme değil, başlangıçtan itibaren ele alınması gereken bir kalite özelliği olarak konumlandırmaktadır.



Şekil 4. Yeşil Fonksiyonel Olmayan Gereksinimler

Çevresel Sürdürülebilirliğin Gereksinim Olarak Tanımlanması

Geleneksel yazılım mühendisliğinde gereksinimler çoğunlukla işlevsellik, performans, güvenlik ve kullanılabilirlik gibi kalite nitelikleri etrafında şekillenmektedir. Yeşil Yazılım Mühendisliği yaklaşımı ise çevresel sürdürülebilirliği bu kalite niteliklerine ek olarak değerlendirmektedir.

Enerji tüketimi ve karbon ayak izinin açıkça ifade edilmediği sistemlerin, farkında olmadan yüksek çevresel maliyetler oluşturabileceğini belirtilmektedir. Bu nedenle, gereksinim analizi aşamasında aşağıdaki türde çevresel gereksinimlerin ve kabul kriterlerinin tanımlanması önerilmektedir [1]:

- Sistem, mümkün olan durumlarda enerji tüketimini minimize edecek biçimde çalışmalıdır.
- Gereksiz veri işleme ve veri transferi önlenmelidir.
- Yüksek enerji tüketimine neden olan işlemler sınırlanabilir veya zamanlanabilir olmalıdır.
- Sistem tasarımı, ölçeklenebilirlik ile enerji verimliliği arasında denge kuracak şekilde planlanmalıdır.

Bu tür gereksinimler, fonksiyonel olmayan gereksinimler kapsamında ele alınmakta ve yazılımın çevresel etkisinin yönetilebilir hâle gelmesini sağlamaktadır. Bu kriterlere bağlı olarak ölçülebilir metrikler ve doğrulanabilir kabul kriterleri tanımlanabilir.

Enerji Farkındalığına Dayalı Gereksinim Belirleme

Yeşil Yazılım Mühendisliği literatüründe öne çıkan temel kavramlardan biri enerji farkındalığıdır. Enerji farkındalığı, yazılımın hangi koşullarda, ne kadar süreyle ve hangi yoğunlukta çalışacağının bilinçli olarak değerlendirilmesini ifade etmektedir.

İncelenen çalışmalar, enerji tüketiminin çoğunlukla tasarım ve uygulama aşamalarında ele alındığını ancak gereksinim aşamasında bu konunun ihmal edildiğini ortaya koymaktadır. Oysa bu aşamada yapılacak değerlendirmeler, yazılımın gereksiz yere sürekli çalışan veya aşırı kaynak tüketen bir yapıya dönüşmesini önleyebilir.

Bu bağlamda, gereksinim analizi sürecinde aşağıdaki soruların sistematik biçimde ele alınması önerilmektedir ([2], [13]):

- Sistemin sürekli çalışması gerçekten gerekli midir?
- Hangi işlemler olay bazlı veya istek bazlı olarak çalıştırılabilir?
- Kullanıcı etkileşimi olmayan zamanlarda sistemin düşük güç moduna geçmesi mümkün müdür?

- Verilerin ne sıklıkla güncellenmesi çevresel açıdan anlamlıdır?

Bu tür sorular, yazılımın gereksinim seviyesinde daha enerji bilinçli bir yapıya kavuşmasını sağlamaktadır.

Gereksinim Önceliklendirmesinde Yeşil Perspektif

Gereksinimlerin önceliklendirilmesi, yazılım projelerinde kaçınılmaz bir süreçtir. Yeşil Yazılım Mühendisliği yaklaşımı, bu önceliklendirme sürecine çevresel etki boyutunun da dâhil edilmesini önermektedir.

Yüksek enerji tüketimine neden olan gereksinimlerin erken aşamada tespit edilmesinin, mimari ve tasarım kararlarını olumlu yönde etkilediği belirtilmektedir. Bu kapsamda, gereksinimler yalnızca iş değeri veya teknik zorluk açısından değil, aynı zamanda potansiyel enerji maliyeti açısından da değerlendirilmelidir.

Sürekli gerçek zamanlı veri işleyen bir özellik, yoğun hesaplama gerektiren raporlama fonksiyonları, arka planda kesintisiz çalışan servisler çevresel açıdan daha yüksek maliyetli gereksinimler olarak değerlendirilebilir. Bu tür gereksinimlerin, alternatif çözümlerle veya sınırlamalarla ele alınması YYM yaklaşımının önemli bir parçasıdır.

Literatürde Önerilen Yaklaşımlar ve Örnekler

Yeşil Yazılım Mühendisliği literatürü, sürdürülebilirliğin gereksinim seviyesinde ele alınmasını özellikle "yaşam döngüsü bakışı" ve "ölçüm temelli yönetim" ile ilişkilendirir ([3], [15]). Endüstri perspektifinde ise sürdürülebilirliğin ürün ve mühendislik liderliği tarafından ölçülebilir ve izlenebilir net hedeflere bağlanması gerektiği vurgulanır [14]. Bu tez bağlamında gereksinim analizi; sürdürülebilirliği NFR setine dahil eden, önceliklendirmede çevresel maliyeti görünür kılan ve sonraki adımlarda kullanılacak ölçüm çerçevesine temel oluşturan aşama olarak ele alınır.

Yazılım Tasarımı ve Mimari

Yazılım tasarımı ve mimari aşaması, sistemin bileşenlerinin nasıl yapılandırılacağını, hangi mimari stillerin kullanılacağını ve bu bileşenlerin nasıl etkileşeceğini belirleyen temel aşamadır. Bu aşamada alınan kararlar, yazılımın yalnızca performansını ve bakım maliyetini değil, aynı zamanda enerji tüketimini ve çevresel etkisini de doğrudan belirlemektedir.

YYM açısından mimari; enerji tüketimini belirleyen en kritik kaldıraçlardan biridir. Yeşil yazılım araştırmaları, sürdürülebilirliğin tasarım ve mimari kararlarla doğrudan şekillendiğini ve ekiplere "enerji verimli seçimler" yaptıracak pratiklere ihtiyaç olduğunu belirtir [13].

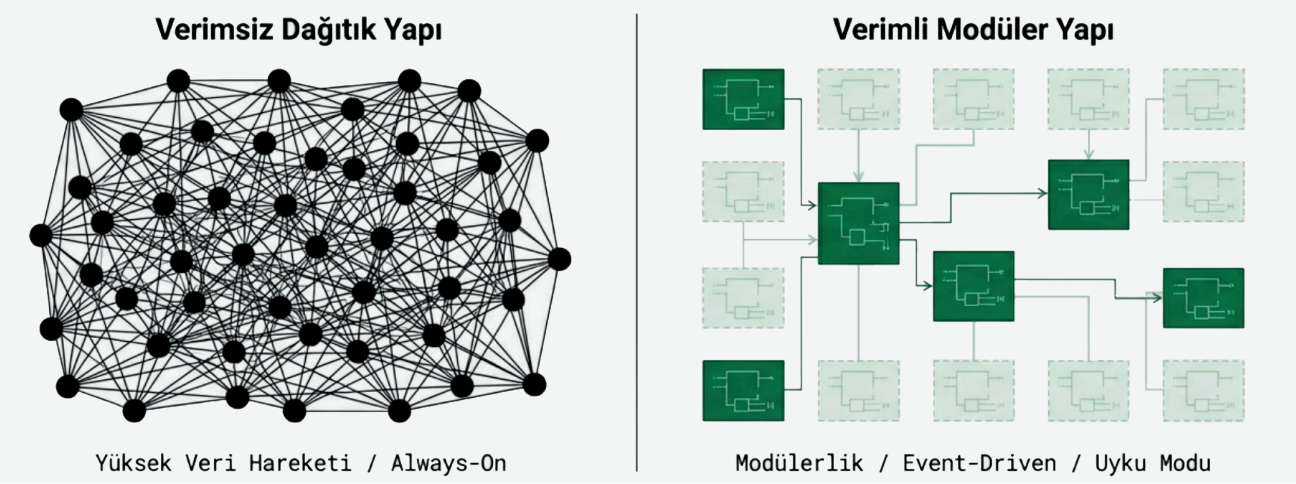
Mimari Kararların Çevresel Etkisi

Yazılım mimarisi; bileşen sayısı, iletişim biçimleri, veri akışı ve dağıtım stratejileri gibi unsurları kapsar. Bu unsurların her biri, sistemin çalışma süresince kullandığı işlemci, bellek, ağ ve depolama kaynaklarını etkilemektedir. Dolayısıyla mimari tasarım, yazılımın çevresel ayak izinin büyük bölümünü şekillendirmektedir.

İncelenen çalışmalarda mimari kararların enerji tüketimi üzerindeki etkisinin çoğu zaman dolaylı ve göz ardı edilen bir faktör olduğunu belirtmektedir ([3], [14]). Şekil 5' de de gösterildiği gibi örneğin aşırı derecede dağıtık bir mimari, yüksek ağ trafiğine ve dolayısıyla artan enerji tüketimine neden olabilmektedir. Bu durum, yalnızca performans açısından değil, çevresel sürdürülebilirlik açısından da değerlendirilmelidir.

Bu bağlamda YYM, mimari tasarım sürecinde aşağıdaki soruların ele alınmasını önermektedir ([13], [15]):

- Gerçekten bu kadar dağıtık bir topolojiye ihtiyaç var mı?
- Servisler arası çağrı sayısı ve payload (taşınan veri) boyutu optimize edilebilir mi?
- Senkron çağrı yerine event-driven (olay temelli) /asenkron model çevresel etkiyi azaltır mı?
- Veri çoğaltma ve cache (ön bellek) stratejisi toplam veri hareketini büyütüyor mu?
- Mimari, enerji tüketiminin izlenmesine olanak tanıyacak biçimde tasarlanmış mıdır?



Şekil 5. Tasarım ve Mimari

Mimari Yapı Seçiminde Yeşil Yaklaşım

Yeşil Yazılım Mühendisliği literatüründe, mimari seçiminin enerji verimliliği üzerinde belirleyici olduğu sıklıkla vurgulanmaktadır. Monolitik, katmanlı, servis tabanlı ve mikroservis mimarileri; enerji tüketimi açısından farklı karakteristikler göstermektedir.

Tek bir mimari stilin her koşulda "en yeşil" çözüm olarak değerlendirilemeyeceği; bunun yerine bağlama duyarlı (context-aware) kararlar alınması gerektiği belirtilmektedir. Bu bağlam; sistemin iş yükü profili, ölçeklenebilirlik gereksinimleri, veri hareketliliği, dağıtım modeli ve çalışacağı altyapının karbon yoğunluğu gibi çevresel ve operasyonel parametreleri kapsamaktadır. Örneğin monolitik mimariler, düşük ağ trafiği nedeniyle bazı senaryolarda daha az enerji tüketebilir. Mikroservis mimarileri, ölçeklenebilirlik avantajı sağlasa da, servisler arası iletişim nedeniyle daha yüksek enerji maliyetleri oluşturabilir. Bu noktada "gereksiz dağıtıklık" kavramına dikkat çekilmekte ve yalnızca teknolojik eğilimlere uyma amacıyla seçilen mimarilerin çevresel açıdan verimsiz olabileceği vurgulanmaktadır.

Modülerlik ve Bileşen Bazlı Tasarım

Yeşil Yazılım Mühendisliği yaklaşımında öne çıkan bir diğer ilke modülerliktir. Modüler mimariler, sistem bileşenlerinin bağımsız olarak geliştirilmesini, çalıştırılmasını ve gerektiğinde kapatılmasını mümkün kılmaktadır. Bu durum, enerji tüketiminin daha alt bileşenlerde kontrol edilmesine olanak tanır.

Modüler sistemlerin özellikle bakım ve evrim aşamalarında enerji açısından avantaj sağladığı belirtilmektedir ([3], [14]). Kullanılmayan veya düşük öncelikli bileşenlerin devre dışı bırakılması, sistemin genel enerji tüketimini azaltabilmektedir.

Bu kapsamda, YYM açısından modüler tasarım gereksiz bileşenlerin sürekli çalışmasının önlenmesi, enerji yoğun modüllerin izole edilerek optimize edilebilmesi, sistem dönüşümü sırasında tüm yapının yeniden çalıştırılmasının gerekmemesi olarak tanımlanabilir.

Veri Akışı ve Mimari Optimizasyon

Yazılım mimarisinde veri akışı, çevresel etki açısından kritik bir faktördür. Büyük veri hacimleriyle çalışan sistemlerde, veri transferi ve veri çoğaltma süreçleri önemli enerji tüketimine neden olmaktadır.

Veri akışının sadeleştirilmesinin, çoğu zaman algoritmik optimizasyondan daha yüksek enerji tasarrufu sağladığı ifade edilmektedir [4]. Bu nedenle mimari tasarımda gereksiz veri kopyalarının önlenmesi, veri sıkıştırma ve toplulaştırma (aggregation) tekniklerinin kullanılması, yerel işlemeyi (edge processing) mümkün kılan mimari yaklaşımların değerlendirilmesi önerilmektedir. Yerel işleme, verinin uç cihazlarda (IoT sensörleri, mobil cihazlar, gateway'ler vb.) ön işleme tabi tutulması ve yalnızca anlamlı ya da özetlenmiş bilginin merkezi sistemlere iletilmesi yaklaşımıdır. Bu tür kararlar, yazılımın yalnızca performansını değil, çevresel sürdürülebilirliğini de artırmaktadır.

Literatürde Yer Alan Mimari Yaklaşımlar ve Örnekler

Literatürde, mimari tasarımın çevresel sürdürülebilirlikle ilişkilendirildiği çeşitli çalışmalar bulunmaktadır. GREENSOFT modeli, mimari tasarımı yazılımın yaşam döngüsündeki en kritik sürdürülebilirlik karar noktalarından biri olarak ele almaktadır [5].

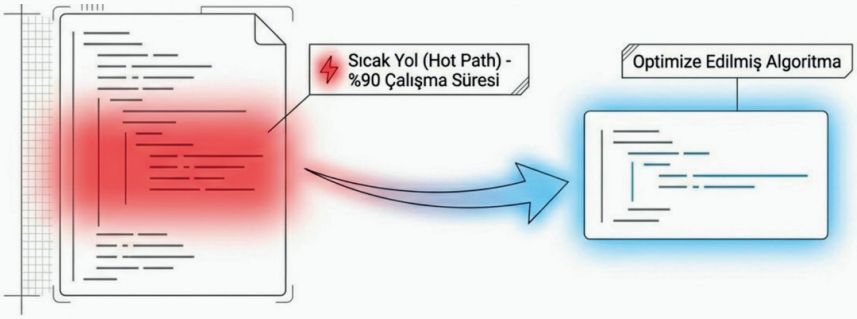
İncelenen bazı çalışmalarda, olay temelli (event-driven) mimarilerin, sürekli çalışan senkron sistemlere kıyasla daha düşük enerji tüketimine sahip olduğu belirtilmiştir [3]. Bu tür mimariler, yalnızca belirli olaylar gerçekleştiğinde aktif hâle gelerek enerji verimliliği sağlamaktadır.

Bir başka çalışmada ise mimari değerlendirme süreçlerine "enerji perspektifinin" eklenilmesi önerilmekte ve mimari kararların yalnızca performans veya ölçeklenebilirlik kriterleriyle değil, karbon yoğunluğu ve enerji orantılılığı açısından da değerlendirilmesi gerektiği savunulmaktadır [16].

Kodlama

Kodlama aşaması, tasarım ve mimari seviyede verilen kararların somutlaştırıldığı, algoritmaların, veri yapıların ve uygulama akışlarının doğrudan hayata geçirildiği aşamadır. Yeşil Yazılım Mühendisliği yaklaşımı açısından bu aşama, çevresel sürdürülebilirliği etkileyen faktörlerin en görünür hale geldiği, aynı zamanda ölçüm ve iyileştirme fırsatlarının en fazla olduğu safhalardan biridir. Ancak literatürde yaygın biçimde vurgulandığı üzere, kod seviyesindeki iyileştirmeler mimari kararların yerine geçirilmeden mimari doğrultuda belirlenmiş sürdürülebilirlik hedeflerini destekleyen tamamlayıcı bir katman olarak ele alınmalıdır ([1], [12]).

Derlenen bazı çalışmalar, yazılımın enerji tüketiminin çoğu zaman "performans optimizasyonu" başlığı altında dolaylı olarak ele alındığını ancak YYM perspektifinin, performansın ötesinde enerji ve karbon etkisini ayrı bir kalite boyutu olarak değerlendirmeyi önerdiğini göstermektedir [3]. Bu nedenle Şekil 6'da da gösterildiği gibi kodlama aşamasında YYM entegrasyonu, enerji farkındalığı, algoritma/veri yapısı seçimleri ve kod seviyesinde kaynak tüketimini azaltan pratikler etrafında yapılandırılabilir.



Şekil 6. Algoritmik Enerji Farkındalığı

Enerji Farkındalığı: Kodun Çalışma Davranışını Görünür Kılma

Yeşil Yazılım Mühendisliği literatüründe "enerji farkındalığı", geliştiricinin kodun kaynak kullanımını yalnızca CPU zamanı açısından değil, işlemci, bellek, disk ve ağ etkileri dahil bütüncül şekilde düşünmesi olarak ele alınmaktadır [12]. Bu yaklaşımda temel amaç, geliştirmenin günlük kararlarını yönlendirecek bir zihinsel model oluşturmaktır.

Kodlama aşamasında enerji farkındalığı aşağıdaki uygulama ilkeleri ile desteklenebilir:

- **Sıcak yol (hot path):** Uygulamanın en sık çalışan fonksiyonları ve en yoğun döngüleri, enerji tüketimi açısından kritik bölgelerdir. Uygulama içerisinde en sık çalıştırılan, en fazla CPU zamanı tüketen veya en yüksek çağrı frekansına sahip kod parçaları "sıcak yol" olarak adlandırılmaktadır. Bu bölgelerde yapılacak küçük iyileştirmeler, toplam etkiye daha fazla katkı sağlar [3].
- **Gereksiz iş yapmama:** YYM rehberleri, "En yeşil işlem yapılmayan işlemdir." yaklaşımını öne çıkarır. Kullanıcıya veya iş sürecine katkısı olmayan loglama, tekrar eden hesaplama, gereksiz veri dönüşümü ve gereksiz seri işlemleri çevresel etkiyi artırır [12].
- **Enerji–performans ayrımı:** Performans iyileştirmesi her zaman enerji iyileştirmesi anlamına gelmeyebilir. Örneğin daha fazla paralellik performansı artırırken toplam enerji tüketimini de artırabilir; bu nedenle enerji farkındalığı, kararların "maliyet–fayda" açısından değerlendirilmesini gerektirir ([1], [3]).

Bu kapsamda, kodlama standartlarında yalnızca performans değil; gereksiz kaynak tüketimi oluşturan kod kalıplarının (anti-pattern) tanımlanması ve bu kalıplardan kaçınılması da önerilmektedir.

Algoritma ve Veri Yapısı Seçimi: Karmaşıklık, Bellek ve Veri Hareketi

Kodlama aşamasında çevresel etkiyi belirleyen en önemli teknik faktörlerden biri, algoritma seçimi ve buna bağlı olarak CPU/bellek kullanımının değişmesidir. Araştırmalar sürdürülebilir yazılım mühendisliğinde klasik kalite özelliklerinin yanında enerji tüketimini etkileyen unsurların tasarım ve uygulama kararlarına dâhil edilmesi gerektiğini belirtmektedir.

Bu bağlamda algoritma seçimi üç ana başlıkta ele alınabilir.

1. **Zaman karmaşıklığı (CPU yoğunluğu)**

   Büyük veri üzerinde çalışan veya sık tetiklenen akışlarda, algoritma karmaşıklığının düşürülmesi enerji tüketimini azaltabilir. Örneğin sıralama gerekmiyorsa sıralamayı kaldırmak, tekrarlı aramaları hash (özet değer) tabanlı yapılara taşımak, gereksiz iç içe döngüleri kaldırmak CPU üzerindeki yükü azaltarak enerji tüketimini düşürebilir. Bu tür yaklaşımlar literatürde "hesaplama maliyetini azaltma" ekseninde YYM ile ilişkilendirilmiştir ([3], [12]).

2. **Bellek karmaşıklığı (RAM baskısı)**

   Bellek baskısı arttıkça çöp toplama (garbage collector), sayfalama (paging) ve cache kaçırma (cache miss) gibi etkiler artabilir. Bu durum, yalnızca performansı değil enerji tüketimini de olumsuz etkileyebilir. Bu nedenle YYM yaklaşımı, bellek açısından daha verimli veri yapılarının tercih edilmesini önerir [12].

3. **Veri hareketi (I/O ve ağ maliyeti)**

   Literatürde, enerji maliyetinin önemli bir kısmının veri hareketinden (ağ / disk I/O) kaynaklanabildiği vurgulanmaktadır. Bu nedenle algoritma seçimi yalnızca CPU açısından değil "veriyi ne kadar taşıdığımız" açısından da değerlendirilmelidir ([3], [12]). Bu durumlar için veri transferini azaltan toplulaştırma (aggregation), gereksiz gidiş - dönüş (round-trip)'leri önleme, uygun önbellekleme (caching) kullanımı, büyük taşınan veri (payload)'leri parçalara ayırma veya sıkıştırma yöntemleri uygulanabilir.

Kod Seviyesinde Yeşil Pratikler

Kod seviyesinde yeşil pratikler, Yeşil Yazılım Mühendisliği'nin "ölç–iyileştir–doğrula" mantığına uygun olarak, kaynak tüketimini düşürmeye yönelik teknik uygulamalar bütünüdür. Araştırmalar yazılımın çevresel etkisini azaltmak için "az iş yapma", "az veri taşıma" ve "az kaynak tüketme" ilkeleri öne çıkmaktadır.

Aşağıdaki pratikler, teorik çerçevede kodlama aşamasına entegre edilebilir:

1. **Gereksiz hesaplamayı azaltma**

   Tekrar eden hesaplamaları önbelleğe almak (memorization / caching), döngüler içinde değişmeyen ifadeleri döngü dışına taşımak, gereksiz format dönüşümlerini (string ↔ json ↔ object) minimize etme gibi pratikler uygulanabilir. Bu pratikler, "işlem sayısını azaltma" ekseninde enerji tüketimini etkileyen faktörler olarak literatürde değerlendirilmektedir.

2. **Kaynak kullanımını sınırlama ve "erken çıkış" (early exit)**

   Filtreleme/validasyon işlemlerini akışın en başında yaparak gereksiz iş yükünü önlemek, kısa devre mantığı (short-circuit) ile koşulların erken sonuçlanmasını sağlamak ve istek/işlem başına maksimum sınırlar (limit) koyma gibi yöntemler uygulanabilir. Bu yaklaşım, YYM'nin "gereksiz iş üretmeme" ilkesiyle uyumludur.

3. **I/O ve ağ trafiğini azaltma**

   Batch işlemeyi teşvik etmek (tek tek çağrı yerine toplu çağrı), veri sorgularında sadece gerekli alanları çekmek (projection), log seviyelerini kontrollü kullanmak (özellikle yüksek trafikli akışlarda) ve veri hareketinin enerji maliyeti literatürde sık tekrar eden durumlardır.

4. **Paralellik ve eşzamanlılığın kontrollü kullanımı**

   Paralel çalıştırma bazı senaryolarda daha kısa sürede bitirmeyi sağlasa da, daha fazla çekirdeğin kullanılıyor olması toplam enerji tüketimini artırabilir. Bu nedenle YYM yaklaşımı, paralelliğin "varsayılan çözüm" olarak değil, ölçüm ve gereksinim bağlamında kontrollü bir karar olarak ele alınmasını önerir.

5. **Platforma duyarlı verimlilik**

   Aynı algoritma farklı platformlarda farklı enerji profiline sahip olabilir. Bu nedenle kodlama aşamasında hedef ortamın (bulut, konteyner, edge, mobil) kısıtları göz önüne alınmalıdır. Bu yaklaşım, YYM'nin bağlam odaklı (context-aware) karar verme anlayışıyla uyumludur. Örneğin, yüksek bellek kullanan bir önbellekleme (in-memory caching) stratejisi bulut ortamında ölçeklenebilirlik açısından uygun olabilirken, mobil veya edge cihazlarda batarya tüketimini hızla artırabilir. Benzer şekilde, paralel işlemeyi yoğun kullanan bir algoritma çok çekirdekli bulut sunucularında verimli çalışırken, sınırlı işlem gücüne sahip IoT cihazlarında gereksiz enerji harcamasına neden olabilir. Bu nedenle enerji verimliliği yalnızca algoritmanın teorik karmaşıklığına değil, çalıştığı donanım ve dağıtım ortamının özelliklerine göre değerlendirilmelidir.

Literatürde Yer Alan Örnek Yaklaşımlar ile İlişkilendirme

Nurmivaara [3], tarafından incelenen çalışmalar, kod düzeyindeki enerji verimliliği araştırmalarının özellikle şu alanlarda yoğunlaştığını göstermektedir:

- Mobil uygulamalarda enerji tüketimi,
- Bulut uygulamalarında kaynak kullanım optimizasyonu,
- IoT/edge senaryolarında veri transferinin azaltılması.

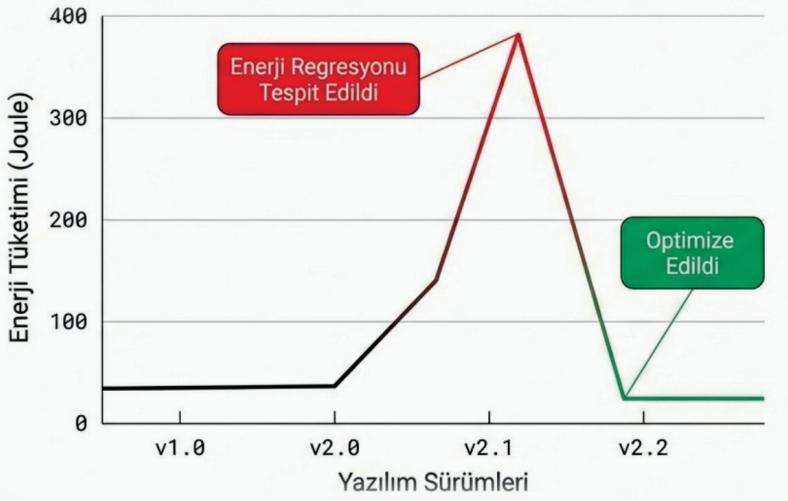
Bu eğilim, kodlama aşamasındaki YYM pratiklerinin çoğunlukla dar bağlamlarda ele alındığını, genel

geçer bir standarttan ziyade bağlamsal rehberler ve teknik öneriler şeklinde geliştiğini göstermektedir. Bu dağınık teknik öneriler "ilkeler" etrafında birleştirilerek geliştiriciler için uygulanabilir bir çerçeve sunmaktadır.

Test ve Doğrulama Aşamasında Yeşil Yazılım Mühendisliği Yaklaşımı

Test ve doğrulama aşaması, yazılımın beklenen işlevsel gereksinimleri karşılayıp karşılamadığının ve kalite niteliklerinin (performans, güvenlik, kullanılabilirlik vb.) doğrulandığı kritik bir aşamadır. Yeşil Yazılım Mühendisliği yaklaşımında ise test süreci yalnızca fonksiyonel doğrulama amacı taşımayarak aynı zamanda yazılımın enerji tüketimi, kaynak kullanımı ve mümkünse karbon etkisi açısından kabul edilebilir sınırlar içinde kalıp kalmadığını görünür kılan bir kontrol mekanizması olarak değerlendirilir. Şekil 7' de de gösterildiği üzere örneğin kontroller sonucu bir sürümde tespit edilen enerji tüketim fazlalığı bir sonraki sürümde iyileştirilebilir.

Nurmivaara [3] tarafından ele alınan çalışmalar, enerji verimliliğinin çoğu zaman tasarım veya kod optimizasyonu aşamalarında ele alındığını, test süreçlerine enerji perspektifinin tutarlı biçimde entegre edilmesinin ise sınırlı kaldığını göstermektedir. Oysa YYM yaklaşımına göre, "ölçemediğini yönetemezsin" ilkesi doğrultusunda, sürdürülebilirlik hedeflerinin korunması için test aşamasında enerjiye duyarlı doğrulama adımları tanımlanmalıdır [12].



Şekil 7. Enerji Regresyonunun Tespiti

Yeşil Kalite Ölçütlerinin Test Kapsamına Alınması

YYM perspektifinde test kapsamı genişletilerek, klasik test türlerine ek olarak enerji ve kaynak odaklı doğrulamalar da ele alınabilir. Bu doğrultuda, test planında aşağıdaki unsurların yer alması önerilmektedir:

- Kritik iş akışlarının kaynak profili (CPU, bellek, I/O, ağ),
- Uzun süreli çalışmada kaynak sızıntısı (memory leak, connection leak) riski,
- Gereksiz tekrarlı işlem, gereksiz veri transferi ve yüksek log üretimi gibi enerji anti-pattern davranışları,
- Yoğunluk altında (yük testinde) sistemin enerji orantılılığı (yük artınca tüketimin nasıl değiştiği).

Araştırmalar enerji tüketiminin çoğu zaman "yük altında" belirginleştiğini ve bu nedenle sürdürülebilirlik hedeflerinin yalnızca birim test düzeyinde değil, senaryo ve yük testleri ile doğrulanması gerektiğini vurgulamaktadır.

Enerji Regresyonu Kavramı ve Regresyon Testlerine Entegrasyon

Yazılım geliştirme süreçlerinde regresyon testleri, yeni değişikliklerin eski fonksiyonları bozmadığını kanıtlamayı amaçlar. YYM yaklaşımı bu kavrama "enerji regresyonu" perspektifini ekler. Yeni bir değişiklik işlevi bozmasa bile, enerji tüketimini veya kaynak kullanımını olumsuz yönde artırmış olabilir.

Bu nedenle, regresyon testleri kapsamında aşağıdaki türde kontrollerin teorik olarak tanımlanması önerilir:

- Kritik endpoint/iş akışları için baz (baseline) süre + kaynak profili referansının tutulması,
- Yeni sürümlerde bu profilden sapmaların izlenmesi,
- Belirli eşiklerin aşılması hâlinde (ör. CPU kullanımında belirgin artış) sürdürülebilirlik açısından riskli değişikliklerin işaretlenmesi.

Calero ve Piattini [1], sürdürülebilirlik yaklaşımının yazılımın tüm yaşam döngüsünde izlenebilir olması gerektiğini; dolayısıyla test sürecinin yalnızca doğrulama değil, sürdürülebilirlik hedeflerini koruma işlevi de görmesi gerektiğini belirtmektedir.

Test Türlerinde Yeşil Yazılım Mühendisliği Perspektifi

YYM'nin test ve doğrulama süreçlerine entegrasyonu, mevcut test türlerini tamamen değiştirmekten ziyade, her bir test türüne çevresel ölçütlerin "ek boyut" olarak dahil edilmesiyle yapılabilir:

a. Birim Testler (Unit Tests)

Birim testlerde doğrudan enerji ölçümü pratikte zor olsa da, enerji tüketimiyle ilişkili davranışları tetikleyen kod kalıpları erken yakalanabilir. Gereksiz döngü ve tekrarlı hesaplama, aşırı nesne üretimi, gereksiz serialization/deserialization işlemlerinden uzak durulması gerekir. Bu yaklaşım, "erken safhada gereksiz iş üretmeme" ilkesini destekler [12].

b. Entegrasyon ve Senaryo Testleri

Entegrasyon testleri, ağ ve I/O yoğun davranışların ortaya çıktığı testler olduğundan YYM açısından kritiktir. Tek bir kullanıcı senaryosunda yapılan çağrı sayısının artması (chatty communication), büyük payload transferleri, veritabanı sorgularında gereksiz alan çekimi (projection eksikliği) gibi durumlar enerji tüketiminin önemli bir kaynağı olan veri hareketi ile doğrudan ilişkilidir ([3], [12]).

c. Performans/Yük Testleri

Performans testleri, YYM perspektifinde aynı zamanda "enerji etkisi görünürleştirme" aracıdır. Yük testlerinde aynı iş yükü altında CPU/bellek/ağ kullanımının izlenmesi, ölçekleme davranışlarının (auto-scaling) kontrol edilmesi gibi yöntemlerle kaynak tüketiminin gereksiz şekilde artıp artmadığının değerlendirilmesi önerilmektedir. Bununla birlikte enerji davranışının daha kapsamlı analiz edilebilmesi için farklı performans test türlerinden de yararlanılabilir. Örneğin yük testleri (load testing) sistemin normal çalışma koşullarındaki enerji tüketimini gözlemlemeyi sağlarken, stres testleri (stress testing) sistem sınırlarının zorlandığı durumlarda enerji tüketiminin nasıl değiştiğini ortaya koyabilir. Uzun süreli dayanıklılık testleri (soak testing) ise sistemin uzun süreli çalışması sırasında kaynak tüketiminde zamanla oluşabilecek artışları veya enerji verimsizliklerini tespit etmeye yardımcı olabilir.

Sistemin enerji orantısallığı (energy proportionality) da doğru analiz edilmelidir. Enerji orantısallığı, iş yükü arttıkça enerji tüketiminin benzer oranda artmasını; düşük iş yükü altında ise sistemin enerji tüketiminin anlamlı biçimde azalmasını ifade etmektedir. Yük ile enerji tüketimi arasında doğrusal veya öngörülebilir bir ilişki göstermelidir.

Test Ortamı ve Süreç Tasarımı: Ölç-İyileştir-Doğrula Döngüsü

Calero ve Moraga [12], rehberinde öne çıkan yaklaşım, sürdürülebilirlik hedeflerinin ölçümle beslenen bir iyileştirme döngüsü ile yönetilmesidir. Bu çerçevede test aşaması, teorik olarak aşağıdaki döngüyle ele alınabilmektedir:

1. Kritik senaryolar belirlenir (yüksek trafik, yüksek hesaplama).
2. Kaynak profili baz alınır (baseline).
3. Değişiklik sonrası profil yeniden gözlemlenir.
4. Sapma varsa neden analizi yapılır.
5. İyileştirme sonrası yeniden doğrulama yapılır.

Nurmivaara [3] tarafından yürütülen çalışmalar, sürdürülebilirlik çalışmalarının önemli bir kısmının "ölçüm zorlukları" nedeniyle sınırlı kaldığını göstermektedir. Bu nedenle, test süreçlerinde hedef, mutlak enerji değerleri yerine, bağlam içinde tekrarlanabilir ve izlenebilir bir karşılaştırma oluşturmak olmalıdır.

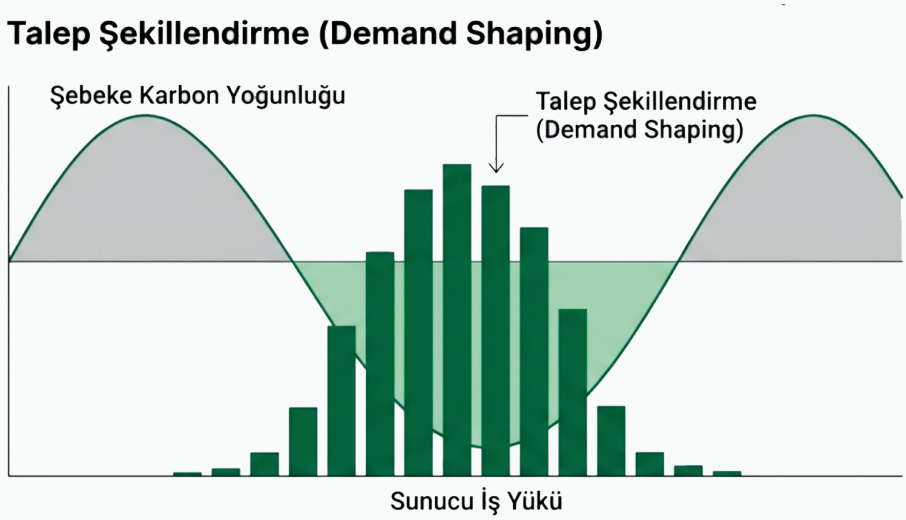
Test ve doğrulama aşaması, Yeşil Yazılım Mühendisliği hedeflerinin sürdürülebilir biçimde korunması açısından kritik bir kontrol adımıdır. Enerji ve kaynak kullanımına ilişkin göstergeler, yazılımın fonksiyonel doğruluğu kadar stratejik hale gelmektedir. YYM yaklaşımı, test süreçlerinin yalnızca "Doğru çalışıyor mu?" sorusuna değil, aynı zamanda "Doğru çalışırken gereksiz kaynak tüketiyor mu?" sorusuna da yanıt vermesini hedefler ([1], [3], [12]).

Bu nedenle, test süreçlerine enerji farkındalığının entegre edilmesi, literatürde ifade edilen standartlaşma eksikliği ve ölçüm zorluklarına rağmen, YYM'nin yazılım yaşam döngüsüne yerleşmesi açısından önemli bir yöntem adımıdır.

Dağıtım (Deployment) ve Çalıştırma (Operasyon) Aşamasında Yeşil Yazılım Mühendisliği Yaklaşımı

Dağıtım ve operasyon aşaması, yazılımın gerçek kullanıcılarla buluştuğu ve çevresel etkinin fiilen ortaya çıktığı aşamadır. Bu aşamada yazılımın enerji tüketimi, yalnızca kodun verimliliğine değil aynı zamanda altyapı seçimine, kaynak tahsisine, ölçekleme davranışına, gözlemlenebilirliğe ve çalışma zamanındaki kullanım desenlerine bağlıdır. Yeşil Yazılım Mühendisliği yaklaşımı, sürdürülebilirlik hedeflerinin uygulama yaşam döngüsünde kalıcı olabilmesi için operasyon aşamasını merkezi bir konuma yerleştirir.

Calero ve Piattini [1], sürdürülebilirliğin yalnızca geliştirme sırasında değil, yazılımın üretim ortamındaki "kullanım ve bakım" evrelerinde de yönetilmesi gereken bir kalite boyutu olduğunu vurgular. Nurmivaara [3] ise literatürde birçok çalışmanın geliştirme ve tasarım odaklı kaldığını, ancak çevresel etkilerin en belirgin olduğu alanlardan birinin üretim/operasyon olduğunu göstermektedir. Bu bağlamda, dağıtım ve operasyon sürecine YYM'nin entegrasyonu enerji ve karbon etkisini azaltacak biçimde "çalışma zamanını" yönetmeyi amaçlamaktadır. Şekil 8'de de gösterildiği gibi bu yaklaşım kapsamında, iş yükünün karbon yoğunluğu düşük zaman dilimlerine kaydırılması anlamına gelen talep şekillendirme (demand shaping) stratejileri önemli bir rol oynamaktadır.



Şekil 8. Dağıtım ve Operasyon

Kaynak Tahsisi ve Kapasite Planlama: Aşırı Tahsisin Önlenmesi

Operasyon aşamasında enerji tüketimini artıran yaygın nedenlerden biri, sistem kaynaklarının ihtiyacın çok üzerinde belirlenip ayrılmasıdır (over-provisioning). YYM yaklaşımı, kapasite planlamasında yalnızca performans ve erişilebilirlik değil enerji verimliliği ve kaynak kullanım oranı gibi ölçütlerin de dikkate alınmasını önerir.

Bu çerçevede teorik olarak uygulanabilecek yöntemler şunlardır:

- **Doğru boyutlama (right-sizing):** Servislerin CPU/bellek limit ve isteklerinin gerçek yük profiline göre ayarlanması.
- **Kapasite hedefleri:** Kaynak kullanım oranlarının kabul edilebilir alt–üst sınırlar içinde tutulması.
- **Yük desenine göre kaynak politikaları:** Gün içi yoğunluk farklılıklarına göre planlı kaynak yönetimi.

Bu yaklaşım, YYM'nin "enerji orantılılığı" ilkesini destekler: Sistem kaynakları, yük arttıkça artmalı; yük azaldıkça da düşmelidir [12].

Ölçekleme Stratejileri: Enerji Orantılı Çalışma Zamanı Davranışı

Bulut ve konteyner tabanlı ortamlarda ölçekleme, performans ve süreklilik açısından kritik olduğu kadar enerji tüketimi açısından da belirleyicidir. Yeşil Yazılım Mühendisliği yaklaşımı ölçeklemeyi, yalnızca hizmet sürekliliğini koruyan bir mekanizma değil, aynı zamanda enerji tüketimini optimize eden bir kontrol aracı olarak ele alır.

Bu kapsamda teorik düzeyde öne çıkan uygulamalar:

- **Yatay ölçeklemenin kontrollü kullanımı:** Gereksiz replika artışları ağ trafiği ve kaynak tüketimini büyütebilir.
- **Otomatik ölçekleme eşiklerinin yeşil perspektifle belirlenmesi:** Yalnızca CPU yüzdesine değil, istek hacmi ve iş çıktısına göre karar mekanizmaları kurulması sağlanmalıdır.
- **Zaman bazlı ölçekleme:** Trafiğin öngörülebilir biçimde değiştiği sistemlerde, belirli saat aralıklarında kaynakların azaltılması sağlanmalıdır.

Nurmivaara [3], literatürde ölçekleme davranışlarının enerji tüketimiyle ilişkilendirildiği çalışmaların arttığını belirtmektedir. Bu da operasyon aşamasının sürdürülebilirlik açısından kritik olduğunu destekler.

Karbon Yoğunluğu ve Çalıştırma Ortamı Seçimi

Yeşil Yazılım Mühendisliği yaklaşımı, aynı enerji tüketiminin farklı bölgelerde farklı karbon etkisi doğurabileceğini vurgular. Bu durum, enerji üretim kaynaklarının bölgesel farklılıklarından kaynaklanır. Calero ve Moraga [12], yazılımların mümkünse düşük karbon yoğunluklu enerji ile çalışan ortamlarda çalıştırılmasının çevresel etkiyi azaltabileceğini belirtir.

Bu doğrultudaki teorik yöntemler aşağıdaki gibidir:

- **Dağıtım bölgesi seçimi:** Uygun durumlarda daha düşük karbon yoğunluklu veri merkezi/bölge tercihi.
- **Zamanlama:** Karbon yoğunluğunun düşük olduğu zaman dilimlerinde enerji yoğun işler (batch) çalıştırma (demand shaping).
- **İş yükü ayrıştırma:** Enerjisi yoğun arka plan işlerini farklı stratejilerle yönetme.

Bu yaklaşım, YYM'nin "karbon farkındalığı" ve "demand shaping" ilkeleriyle uyumludur [12].

Gözlemlenebilirlik: Operasyonda Ölçüm ve Geri Besleme

Operasyon aşamasında sürdürülebilirlik hedeflerinin yönetilebilmesi için ölçüm ve izleme kritik önem taşımaktadır. Yeşil Yazılım Mühendisliği literatüründe ölçüm zorlukları sıkça vurgulanmakla birlikte pratikte izlenebilir göstergeler üzerinden sürekli iyileştirme yaklaşımı önerilmektedir ([3], [12]).

Bu bağlamda teorik olarak izlenebilecek metrik sınıfları aşağıdaki gibidir:

- **Kaynak metrikleri:** CPU, bellek, disk I/O, ağ trafiği,
- **İş çıktısı metrikleri:** İstek/saniye, işlenen kayıt sayısı, tamamlanan görev sayısı,
- **Verimlilik oranları:** İş çıktısı başına kaynak tüketimi (energy/resource efficiency),
- **Davranışsal göstergeler:** Boşta bekleme süreleri, aşırı log üretimi, retry oranları.

Bununla birlikte izleme sistemlerinin kendisi de enerji ve depolama maliyeti üretebilen bileşenlerdir. Özellikle log, metrik ve iz (trace) verilerinin yüksek hacimde üretilmesi; depolama alanı, ağ trafiği ve analiz altyapısı üzerinde ek yük oluşturmaktadır. Bu nedenle log seviyelerinin (debug, info, warn vb.) ortam bazlı yapılandırılması, üretim ortamında gereksiz ayrıntılı logların kapatılması, tüm verilerin sürekli saklanması yerine örnekleme (sampling) tekniklerinin uygulanması ve saklama sürelerinin (retention policy) optimize edilmesi önerilmektedir. Bu yaklaşım, görünürlük ile kaynak verimliliği arasında denge kurmayı amaçlamaktadır.

Calero ve Piattini [1], sürdürülebilirliğin mühendislik disiplini haline gelebilmesi için ölçümle desteklenmesi gerektiğini vurgular. Bu nedenle operasyon, YYM açısından sürekli geri besleme döngüsünün merkezidir.

Sürüm yönetimi ve dağıtım (deployment) stratejileri de operasyonel sürdürülebilirlik açısından değerlendirilmelidir. Her yeni dağıtım; derleme (build), konteyner imaj üretimi (container image build), test çalıştırma (test execution), yazılım çıktılarının saklanması (artifact storage), imaj kayıt deposuna aktarım (registry transfer) ve sürümün ortama yaygınlaştırılması (rollout deployment) süreçleri nedeniyle ek işlem ve ağ trafiği üretmektedir. Bu nedenle gereksiz veya sık aralıklı dağıtımların toplam enerji maliyeti göz ardı edilmemelidir. Sürüm sıklığının iş değeri ile dengelenmesi, yalnızca anlamlı değişiklikler için dağıtım yapılması ve pipeline süreçlerinin optimize edilmesi önerilmektedir. Ayrıca kontrollü dağıtımlar; hatalı sürümlerin tüm sisteme yayılmasını önleyerek geri alma (rollback) maliyetini ve gereksiz yeniden dağıtımları azaltabilmektedir. Bu yaklaşım, operasyonel kararlarda çevresel etkiyi de dikkate alan bağlam odaklı bir yönetim anlayışı ile uyumludur.

Bakım ve Evrim Aşamasında Yeşil Yazılım Mühendisliği Yaklaşımı

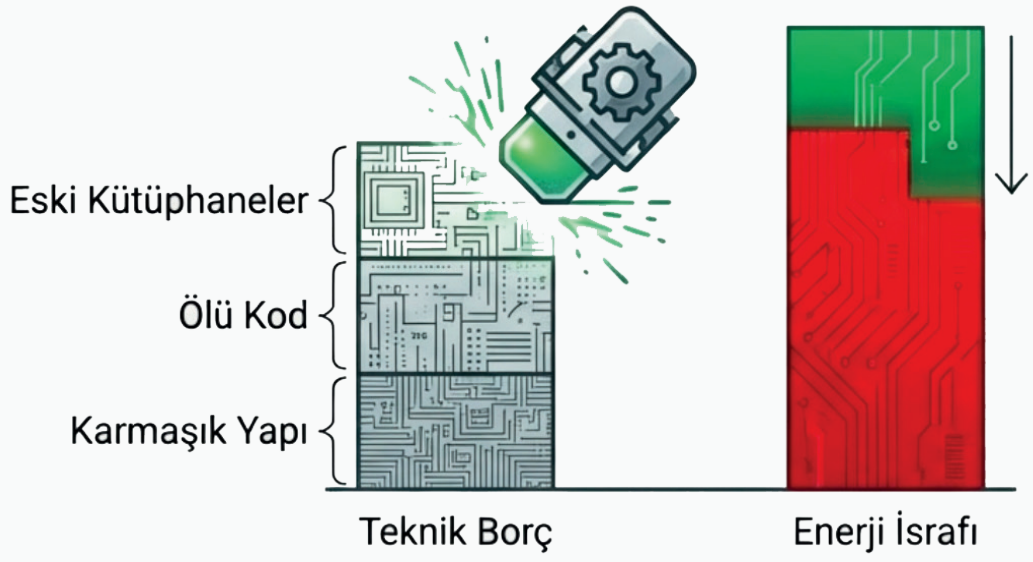
Yazılım yaşam döngüsünün bakım ve evrim aşaması, sistemin üretim ortamında çalışmaya başladıktan sonra değişen ihtiyaçlara uyum sağlaması, hataların giderilmesi, performans iyileştirmeleri, güvenlik güncellemeleri ve yeni özelliklerin eklenmesi süreçlerini kapsar. Bu aşama, yazılımın toplam yaşam süresinin önemli bir kısmını oluşturur ve çevresel sürdürülebilirlik açısından kritik önem taşımaktadır. Yeşil Yazılım Mühendisliği yaklaşımı, sürdürülebilirliğin tek seferlik bir optimizasyon faaliyeti değil, yazılımın yaşamı boyunca korunması gereken **sürekli bir mühendislik hedefi** olduğunu vurgular ([1], [12]).

Nurmivaara [3] tarafından yapılan çalışmalarda, sürdürülebilirlik çalışmalarının önemli bölümünün "tasarım ve geliştirme" aşamasında yoğunlaştığı; bakım ve evrim aşamasında sürdürülebilirliği yöneten sistematik yaklaşımların daha sınırlı ele alındığı görülmektedir. Oysa üretim ortamında sistem büyüdükçe bileşen sayısı, veri hacmi, kullanım yoğunluğu ve teknik borç artmakta, bu durum enerji tüketimini ve çevresel etkiyi yükseltebilmektedir. Bu nedenle bakım aşaması, YYM açısından "sürdürülebilirliğin korunması ve gerilemenin önlenmesi" amacıyla ele alınmalıdır.

Teknik Borç ve Sürdürülebilirlik

Bakım aşamasında yazılım sistemlerinin en önemli risklerinden biri teknik borcun artmasıdır. Teknik borç, kısa vadeli kazanımlar uğruna ertelenen tasarım veya kod kalitesi iyileştirmelerinin uzun vadede maliyet oluşturmasıdır. Yeşil Yazılım Mühendisliği perspektifinde teknik borç yalnızca bakım maliyetini artırmaz aynı zamanda gereksiz kaynak tüketimi nedeniyle çevresel etkiyi de büyütebilir.

Sürdürülebilir yazılım hedeflerinin korunması için "gereksiz iş üretmeme" ve "veriyi gereksiz taşımama" ilkelerinin yaşam boyu uygulanması gerektiğini vurgulanmaktadır. Teknik borç arttıkça tekrar eden hesaplamalar, gereksiz çağrılar, aşırı log üretimi, karmaşık akışlar ve gereksiz bağımlılıklar gibi enerji tüketimini artıran davranışlar sistemin doğal parçası haline gelebilir. Şekil 9'da da gösterildiği gibi zaman içerisinde kullanım dışı kalmış veya iş değeri üretmeyen miras alınan özellikler (legacy features) de bakım maliyetinin yanı sıra sürekli işlem, depolama ve test yükü oluşturarak gizli bir enerji maliyeti yaratabilir. Özellikle geriye dönük uyumluluk (backward compatibility) amacıyla sistemde tutulan ancak aktif olarak kullanılmayan modüller, her sürümde derleme, test ve dağıtım süreçlerine dahil edilerek gereksiz kaynak tüketimine yol açabilir. Bu nedenle bakım sürecinde teknik borç yönetimi, aynı zamanda sürdürülebilirlik yönetimi olarak değerlendirilmelidir.



Şekil 9. Teknik Borç

Yeniden Düzenleme ve Süreç İyileştirme

Bakım ve evrim aşamasında Yeşil Yazılım Mühendisliği entegrasyonu için en etkili yöntemlerden biri, yeniden düzenleme (refactoring) faaliyetlerinin sürdürülebilirlik hedefleriyle ilişkilendirilmesidir. Yeniden düzenleme sistemin davranışını değiştirmeden iç yapısını iyileştirmeyi amaçlar. Bu kapsamda yeşil düzenleme yaklaşımı, aşağıdaki hedeflerle kurgulanabilir:

- Daha az kaynak tüketen kod yapıları oluşturmak,
- Veri akışını sadeleştirmek ve gereksiz dönüşümleri azaltmak,
- Modüller arası gereksiz bağımlılıkları kırmak,
- I/O ve ağ trafiği oluşturan akışları optimize etmek.

Nurmivaara [3], literatürde sürdürülebilirlik hedeflerinin çoğunlukla "optimizasyon" başlığında ele alındığını, bakım döneminde bu optimizasyonların tekrar edilmesi ve süreklilik kazanmasının alanın önemli bir ihtiyacı olduğunu göstermektedir. Bu nedenle bakım aşamasında yeniden düzenleme, yalnızca kod temizliği değil, çevresel etkiyi azaltmaya yönelik planlı bir faaliyet olarak tasarlanmalıdır.

Sürüm Yönetimi ve Enerji Regresyonunun Önlenmesi

Bakım aşamasının bir diğer temel boyutu sürüm yönetimidir. Yazılım sistemleri yeni özellikler kazandıkça veya altyapı güncellendikçe, enerji tüketiminde istenmeyen artışlar oluşabilir. Bu durum, önceki aşamalarda tanımlanan "enerji regresyonu" kavramının bakım döneminde de geçerli olduğunu göstermektedir.

Yeşil Yazılım Mühendisliği yaklaşımı, bakım döneminde aşağıdaki stratejilerin teorik olarak uygulanmasını önermektedir:

- Kritik iş akışları için kaynak profillerinin sürümler arasında izlenmesi,
- Yeni sürümde gözlenen belirgin kaynak artışlarının kök neden analizine konu edilmesi,
- Gereksiz genişleyen loglama maliyetlerinin kontrol edilmesi,
- Sistem büyüdükçe artan veri hacmine karşı arşivleme/yaşam döngüsü yönetimi stratejileri uygulanması.

Calero ve Morega [12], sürdürülebilirliğin ölçümle beslenen bir döngü olduğunu vurguladığından, bakım aşamasında sürüm bazlı karşılaştırmalar sürdürülebilirliğin sürekliliği açısından anlamlıdır.

Bağımlılık Yönetimi ve Teknoloji Evrimi

Bakım aşamasında çevresel etkiyi etkileyen önemli konulardan biri, üçüncü parti kütüphaneler ve platform bağımlılıklarıdır. Zamanla büyüyen bağımlılık ağları, gereksiz paket yükü, artan çalışma zamanı maliyeti ve daha fazla bellek/CPU kullanımı oluşturabilir.

Bu nedenle Yeşil Yazılım Mühendisliği perspektifinde bakım sürecinde gereksiz bağımlılıkların kaldırılması, daha hafif alternatiflerin değerlendirilmesi, versiyon güncellemelerinin performans/enerji etkisi açısından göz önüne alınması önerilmektedir. Bu yaklaşım, sürdürülebilirliği yalnızca "kod yazma" anına değil, teknolojik evrimin tüm adımlarına yaymaktadır.

Literatürde Yer Alan Örnek Yaklaşımlar ile İlişkilendirme

Calero ve Piattini [1], sürdürülebilirliği çok boyutlu bir kalite özelliği olarak ele alırken, bakım döneminde yazılımın teknik sürdürülebilirliğinin korunmasının çevresel sürdürülebilirlikle bağlantılı olduğunu belirtmektedir. Sistem bakımı zayıfladıkça, kaynak tüketimi kontrol dışı kalabilir.

Nurmivaara [3], alanın standardizasyon eksikliği nedeniyle bakım döneminde sürdürülebilirliği yönetecek ortak pratiklerin sınırlı kaldığını ancak literatürde giderek artan biçimde "sürdürülebilirliğin yaşam boyu yönetimi" temasının güçlendiğini göstermektedir.

Calero ve Piattini [12] ise sürdürülebilirliği "sürekli bir optimizasyon disiplini" olarak konumlandırarak, bakım aşamasında ölçüm, yeniden değerlendirme ve sürekli iyileştirme döngüsünün sürdürülmesini önermektedir.

Bu çerçeve, yapay zekâ destekli karar mekanizmalarının sürdürülebilirlik yönetimine entegre edilmesi için uygun bir zemin sunmaktadır. Yapay zekâ; sistem davranışlarını analiz ederek enerji tüketim kalıplarını tespit edebilir, anomali algılama (anomaly detection) ile gereksiz kaynak kullanımını belirleyebilir ve otomatik optimizasyon önerileri üretebilir. Özellikle operasyon aşamasında iş yükü tahmini (workload forecasting), dinamik kaynak tahsisi (dynamic resource allocation) ve otomatik ölçekleme (auto-scaling optimization) gibi alanlarda makine öğrenmesi temelli yaklaşımlar, enerji orantısallığını iyileştirmeye katkı sağlayabilir.

Ayrıca kod düzeyinde yapay zekâ destekli analiz araçları; tekrarlayan hesaplamaları, yüksek frekanslı sıcak yol (hot path) bölgelerini veya gereksiz veri hareketlerini tespit ederek geliştiricilere sürdürülebilirlik odaklı kod düzenleme (refactoring) önerileri sunabilir. Böylece sürdürülebilirlik, yalnızca manuel ölçüm ve değerlendirme sürecine değil, sürekli öğrenen ve uyarlanan bir karar destek sistemine dönüştürülebilir.

Yeşil Yazılım Mühendisliğinde Bulut Teknolojiler

Dijital dönüşüm süreçleriyle birlikte bulut bilişim ve yapay zekâ teknolojileri, yazılım sistemlerinin temel bileşenleri haline gelmiştir. Bununla birlikte bu teknolojiler, enerji tüketimi ve karbon emisyonları açısından önemli çevresel etkiler üretmektedir. Yeşil yazılım mühendisliği (YYM), bu teknolojilerin sürdürülebilirlik hedefleriyle uyumlu şekilde tasarlanmasını ve işletilmesini amaçlayan bütüncül bir yaklaşım sunmaktadır.

Bulut bilişim altyapıları, ölçeklenebilirlik ve esneklik sağlamakla birlikte, veri merkezlerinin artan enerji tüketimi nedeniyle küresel karbon emisyonlarına önemli katkı sağlamaktadır. Literatürde, bilgi ve iletişim teknolojileri sektörünün küresel emisyonlardaki payının giderek arttığı ve veri merkezlerinin elektrik tüketiminin küresel ölçekte anlamlı seviyelere ulaştığı belirtilmektedir [4].

Bhat ve Saha'ya göre yeşil yazılım mühendisliği, sürdürülebilirliği güvenlik ve güvenilirlik gibi temel bir kalite özelliği olarak ele almalı ve "sustainability-by-design" yaklaşımını benimsemelidir [13]. Bu yaklaşım, bulut ortamında aşağıdaki mimari ve operasyonel kararların karbon etkisini dikkate almayı gerektirmektedir:

- Düşük karbon yoğunluğuna sahip veri merkezi bölgelerinin seçilmesi,
- İş yüklerinin zamansal ve mekânsal olarak karbon yoğunluğuna göre kaydırılması,
- Otomatik kaynak ölçekleme (autoscaling) ve doğru boyutlandırma (rightsizing),
- Sunucusuz ve konteyner tabanlı mimarilerin tercih edilmesi.

Özellikle otonom iş yükü optimizasyonu (autonomous workload optimization - AWO) araçları, performans gereksinimlerini koruyarak kaynak kullanımını minimize edebilmekte ve böylece enerji tüketimini azaltmaktadır [17]. Bu araçlar, dinamik kaynak tahsisi yoluyla aşırı provizyonu önleyerek hem maliyet hem de karbon emisyonu azaltımı sağlayabilmektedir. Dolayısıyla bulut mimarisi kararları, yalnızca performans ve maliyet değil, karbon verimliliği açısından da değerlendirilmelidir.

Yeşil Yazılım Mühendisliğinde Yapay Zekâ

Yapay zekâ sistemleri, özellikle büyük dil modelleri ve üretken yapay zekâ uygulamaları, yüksek hesaplama gücü gereksinimleri nedeniyle enerji tüketimini artırmaktadır. Gartner araştırmaları, Generative AI'nın yazılım sistemlerinin karbon ayak izini önemli ölçüde artırabileceğini ve bu nedenle sürdürülebilir yazılım mühendisliği uygulamalarının zorunlu hale geldiğini vurgulamaktadır [15].

Yapay zekâ sistemlerinde karbon etkisi iki temel aşamada ortaya çıkmaktadır:

- Model eğitimi (training),
- Model çalıştırma (inference).

Özellikle büyük ölçekli modellerin GPU kümeleri üzerinde eğitilmesi, yüksek enerji tüketimi ve dolaylı karbon emisyonu üretmektedir. Bu durum, YYM açısından aşağıdaki kritik soruları gündeme getirmektedir:

- Daha küçük ve görev odaklı modeller kullanılabilir mi?
- Modelin yeniden eğitilmesi (fine-tuning) yerine, bilgi getirme temelli (retrieval-based) mimariler tercih edilebilir mi?
- Model eğitimi düşük karbon yoğunluklu zaman dilimlerinde gerçekleştirilebilir mi?
- Sürekli inference gerektirmeyen alternatif mimari tasarımlar mümkün müdür?

Gartner raporları, karbon verimliliği (carbon efficiency) ile karbon farkındalığını (carbon awareness) birlikte ele almanın gerekliliğini vurgulamaktadır [13]. Bu bağlamda AI sistemlerinin tasarımında enerji tüketimi, performans ve doğruluk gibi metriklerle birlikte değerlendirilmelidir.

Sonuç ve Öneriler

Bu çalışma, yazılım mühendisliği disiplininde çevresel sürdürülebilirlik bilincinin yerleşmesi amacıyla geliştirilen Yeşil Yazılım Mühendisliği yaklaşımını temel boyutları ile incelemiştir. Çalışma kapsamında, literatürde yer alan modeller, yöntemler ve metrikler sistematik biçimde analiz edilmiş, disiplinin kavramsal çerçevesi, metodolojik eğilimleri ve karşılaşılan temel eksiklikler ortaya konulmuştur.

Elde edilen bulgular, YYM'nin henüz olgunlaşma sürecinde olan bir disiplin olduğunu, ancak yazılım mühendisliği uygulamaları açısından stratejik öneme sahip olduğunu göstermektedir. 2000'li yıllardan bu yana sürdürülebilir bilişim kavramı öncelikle donanım odaklı çalışmalarda ele alınmış, yazılım bileşenlerinin çevresel etkileri son on yılda sistematik olarak araştırılmaya başlanmıştır.

Bu dönüşüm, yazılımın yalnızca işlevsel veya performans temelli bir varlık olmadığını, aynı zamanda çevresel etkileri bulunan bir dijital ürün olduğunu ortaya koymaktadır. Dolayısıyla yazılım sistemlerinin tasarımında ve bakımında enerji verimliliği, karbon ayak izi, ağ trafiği ve kaynak kullanımı gibi ölçütlerin dikkate alınması sürdürülebilir kalkınma hedefleriyle doğrudan ilişkilidir.

Yeşil Yazılım Mühendisliği alanı çok boyutlu bir yapıya sahiptir ve gelecekteki araştırmaların disiplinler arası yaklaşımlar içermesi önemlidir. Bu doğrultuda aşağıdaki araştırma konuları önerilmektedir:

- **Enerji Verimli Yazılım Metriklerinin Standartlaştırılması:** Yazılım bileşenlerinin enerji tüketimini ölçmek için açık kaynaklı bir metrik standardının geliştirilmesi,
- **Makine Öğrenmesi Tabanlı Enerji Tahmini:** Yazılımın çalışma sürecinde enerji tüketimini önceden tahmin eden modellerin oluşturulması,
- **Bulut ve Mikroservis Mimarilerinde Karbon Optimizasyonu:** Uygulamaların coğrafi olarak karbon yoğunluğu düşük bölgelerde çalıştırılmasını sağlayan dinamik yönlendirme algoritmalarının geliştirilmesi,
- **Enerji farkındalıklı (Green-Aware) CI/CD Süreçleri:** Sürekli entegrasyon (CI) ve dağıtım (CD) süreçlerine enerji verimliliği ölçümlerinin otomatik dâhil edilmesi,
- **Kamu Yazılımlarında Pilot Uygulamalar:** TÜBİTAK ve yerel yönetim projelerinde yeşil yazılım göstergelerinin test edilmesi.

Bu adımlar, yazılım mühendisliği disiplininde çevresel farkındalığın yalnızca ek bir kriter değil, tasarımın doğrudan bir parçası haline gelmesi gerektiğini ortaya koymaktadır.

Bu çalışmadan elde edilen sonuçlar, yalnızca akademik düzeyde değil, kamu kurumları ve teknoloji odaklı kuruluşlar açısından da uygulanabilir öneriler sunmaktadır. Özellikle TÜBİTAK BİLGEM gibi yüksek teknolojili projeler yürüten kurumlar için Yeşil Yazılım Mühendisliği yaklaşımı stratejik bir fırsat alanı oluşturmaktadır.

Aşağıda hem kurumsal hem de ulusal düzeyde öneriler yer almaktadır.

Kurumsal Düzeyde Öneriler

1. **Yeşil kodlama rehberi geliştirilmesi:** Yazılım ekipleri için kodlama standartlarının enerji verimliliği kriterlerini içerecek biçimde genişletilmesi.
2. **Sürdürülebilirlik eğitimleri:** Geliştirici ve teknik liderler için yeşil yazılım farkındalığına yönelik eğitim modüllerinin düzenlenmesi.
3. **Enerji izleme araçlarının kullanımı:** Geliştirme ve test ortamlarında enerji tüketimini ölçen araçların (örneğin PowerAPI, GreenFrame) entegrasyonu.
4. **Yeşil mimari değerlendirme adımı:** Mimari gözden geçirme süreçlerine "Green Review" aşaması eklenerek enerji yoğun bileşenlerin erken tespit edilmesi.

5. **Kurumsal karbon raporlaması:** Yazılım projelerinin enerji tüketimi ve karbon emisyonlarının yıllık kurumsal sürdürülebilirlik raporlarına dâhil edilmesi.

Ulusal Düzeyde Öneriler

1. **Yeşil yazılım sertifikasyonu:** Türkiye'de geliştirilen yazılımlara çevresel etki düzeyine göre "Yeşil Etiket" verilmesini sağlayacak bir sertifikasyon sistemi oluşturulabilir.
2. **Ulusal sürdürülebilir yazılım politikası:** Bilgi Teknolojileri ve İletişim Kurumu (BTK) ve TÜBİTAK koordinasyonunda, enerji verimliliğini teşvik eden yazılım politikaları geliştirilebilir.
3. **Akademi–Sanayi iş birliği:** Üniversitelerde yürütülen yeşil bilişim araştırmalarının sanayi projelerine entegre edilmesi sağlanabilir.
4. **Açık veri platformu:** Yazılım projelerinin enerji profillerini karşılaştırmalı olarak sunan açık bir veri tabanı oluşturularak araştırmacılar için ortak bir ölçüm altyapısı sağlanabilir.
5. **Ar-Ge teşvikleri:** Yeşil yazılım teknolojileri geliştiren girişim ve firmalar için özel teşvik mekanizmaları tasarlanabilir.

Bu öneriler, Türkiye'nin dijital dönüşüm sürecinde enerji verimliliğini yalnızca donanımda değil, yazılım düzeyinde de stratejik bir unsur haline getirmeye katkı sağlayacaktır.

Kaynakça

[1] C. Calero,M. Piattini, **Green in software engineering**. Springer, 2015.

[2] J. Koomey, **Growth in data center electricity use 2005 to 2010.** Oakland, CA: Analytics Press, 2011.

[3] S. Nurmivaara, **"Green in software engineering: A systematic literature review,"** Yüksek Lisans Tezi, Helsinki Üniversitesi, 2023.

[4] Rapaic, M. (2021), **Green software engineering: Developer guide.** Fuller Fellowship.

[5] S. Naumann, M. Dick, E. Kern ve T. Johann, **"The GREENSOFT model: A reference model for green and sustainable software and its engineering,"** Sustainable Computing: Informatics and Systems, cilt 1, no. 4, ss. 294–304, 2011.

[6] N. Rashid vd., **"Green-Agile Maturity Model: An evaluation framework for global software development vendors,"** IEEE Access, cilt 9, ss. 71868–71886, 2021.

[7] J.-M. Horcas, M. Pinto, L. Fuentes ve N. Gámez, **"Self-adaptive energy-efficient applications: The HADAS developing approach,"** 2017 IEEE 15th International Conference on Dependable, Autonomic and Secure Computing, 15th International Conference on Pervasive Intelligence and Computing, 3rd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTec), ss. 140, 2017.

[8] M. Salam ve S. U. Khan, **"Green software multisourcing readiness model (GSM-RM) from vendor's perspective,"** Science International (Lahore), cilt 26, ss. 1421–1424, 2014.

[9] AWS. (2023), AWS customer carbon footprint tool. Erişim Adresi: https://aws.amazon.com/aws-cost-management/aws-customer-carbon-footprint-tool/, Erişim Zamanı : 09 Ocak 2026.

[10] J. Herschmann vd., **"Top strategic trends in software engineering for 2025,"** Gartner, Inc., 2025.

[11] D. Micko ve A. Kapoor, **"Future of software engineering 2030: The impact of AI,"** Gartner, Inc., 2025.

[12] Calero, C. ve Moraga, M. A. (2021). **Software sustainability: Concepts, practices and tools.** Springer.

[13] M. Bhat ve M. Saha, **"Innovation insight for using green software engineering to drive sustainability,"** Gartner, Inc., Rapor No: G00789096, 13 Şubat 2025.

[14] Freed, M., Bielinska, S., Buckley, C., Coptu, A., Yilmaz, M., Messnarz, R., & Clarke, P. M. (2023). **An investigation of green software engineering**. Dublin City University. https://doras.dcu.ie/29087/1/GreenSoftwareEngineeringCameraReady.pdf adresinden erişildi.

[15] M. Driver, **"Emerging tech: Align software engineering with sustainability goals,"** Gartner, Inc., Rapor No: G00812221, 21 Ocak 2025.

[16] Hindle, **"Green mining: A methodology of relating software change and configuration to power consumption,"** 2012 9th IEEE Working Conference on Mining Software Repositories (MSR), ss. 78–87, 2012. https://doi.org/10.1109/MSR.2012.6224301.

[17] Prasad, P., Ennaciri, H. ve Bhat, M. (19 Ağustos 2024). **Innovation insight: Autonomous workload optimization (ID G00797947)**. Gartner.

Kısaltmalar

Kısaltmalar	Açıklama
TÜBİTAK	Türkiye Bilimsel ve Teknolojik Araştırma Kurumu
BİLGEM	Bilişim ve Bilgi Güvenliği İleri Teknolojiler Araştırma Merkezi
YTE	Yazılım Teknolojileri Araştırma Enstitüsü
AWO	Autonomous Workload Optimization (Otonom İş Yükü Optimizasyonu)
BT	Bilgi Teknolojileri
BTK	Bilgi Teknolojileri ve İletişim Kurumu
CPU	Central Processing Unit (Merkezi İşlem Birimi)
ICT	Information and Communication Technology (Bilgi ve İletişim Teknolojileri)
IoT	Internet of Things (Nesnelerin İnterneti)
IT	Information Technology (Bilgi Teknolojileri)
YYM	Yeşil Yazılım Mühendisliği
GAMM	Green - Agile Maturity Model
GSMRM	Green Software Multi – Sourcing Readiness Model

Şekiller ve Tablolar

Serideki tüm şekiller NotebokLM (2026 Güncellemesi) ile üretilmiştir.
Erişim Adresi: https://notebooklm.google.com/, Erişim Zamanı: 13 Şubat 2026.

İşçi Blokları Mahallesi Muhsin Yazıcıoğlu Caddesi No:51/C 06530 Çankaya/ANKARA

+90 (312) 289 92 22 - yte.bilgi@tubitak.gov.tr

TÜBİTAK - BİLGEM Yazılım Teknolojileri Araştırma Enstitüsü (YTE)