Bilgisayar Bilimleri ve Mühendisliği Dergisi

Cilt-16
Sayı-2
2023

Bilimsel Yazılım Geliştirme Süreçleri için bir Yazılım Mühendisliği Yetkinlik Envanteri

A Software Engineering Competency Inventory for Scientific Software Development Processes

Bilge SAY
Atılım Üniversitesi
Yazılım Mühendisliği Bölümü
Ankara, Türkiye
bilge.say@atilim.edu.tr
ORCID: 0000-0001-9276-729X

Öz

Araştırma amaçlı bilimsel yazılım geliştirme süreçleri, yazılım yaşam döngüsü açısından hem süreç hem girdi ve çıktılar bakımından iş veya endüstri amaçlı yazılımlardan bazı farklılıklar göstermektedir. Bilimsel yazılım geliştiricilerin, kod yazma deneyimi olsa da yapısal ve güncel yazılım mühendisliği yetkinliklerinin olmama olasılığı, bilimsel yazılımların gereksinimleri karşılaması ve sürdürülebilirliği açısından sorun yaratabilmektedir. Bu çalışmada literatürde bilimsel yazılım geliştirme alanında gereksinim duyulduğu belirtilen pratikler ile temelde IEEE Yazılım Mühendisliği Yetkinlik Modeli'nin (IEEE's Software Engineering Competency Model -SWECOM) yetkinlikleri eşleştirilmiş; ve bilimsel yazılım geliştirme açısından en gereksinim duyulan yazılım mühendisliği pratikleri belirlenmiştir. Yapılan sıklık analiziyle özellikle yazılım tasarımı ve yapımı sırasındaki detaylı tasarım ve planlama yetkinliklerine ihtiyaç duyulduğu belirlenmiştir. Üretilen envanter, Ar-Ge destekleri çerçevesinde eğitim programları geliştirmek ve iyileştirmek için kullanılabilecektir.

Anahtar sözcükler: Yazılım Mühendisliği Yetkinlikleri, Bilimsel Yazılım Geliştirme, SWECOM

Abstract

Scientific software development processes display some differences from business/industry-aimed software in terms of both processes and input/outputs. Although scientific software developers might have some coding experience, the fact that they may not possess formal and up-to-date software engineering competencies can affect negatively the scientific software produced in terms of meeting the requirements and sustainability. A matching inventory has been created in the current work, pairing requirements for software engineering practices documented in scientific software development literature with competencies mostly from IEEE's Software Engineering Competency Model (SWECOM). Frequencies of the pairings have shown especially a need for software engineering competencies in software design and construction including detailed design and planning. The inventory may be used to generate and improve training programs within research and development support.

Keywords: Software Engineering Competencies, Scientific Software Development, SWECOM

Gönderme, düzeltme ve kabul tarihi: 08.08.2022 - 01.03.2023 – 20.06.2023
Makale türü: Araştırma

1. Bilimsel Yazılım Geliştirmede Yazılım Mühendisliği Yetkinlik Gereksinimlerine Giriş

Bilimsel yazılım geliştirme (araştırmalarda kullanılmak üzere yazılım geliştirme), endüstriyel yazılım geliştirme sürecinden bazı farklı özellikler gösterir. Örneğin, geliştirilen yazılım bir kuramın geçerliğinin sınanması veya modellenmesi hedefini taşıyorsa, araştırma sonuçları geldikçe değişim geçirecektir ve gereksinimlerin en baştan tam olarak bilinmesi zor olacaktır. Kuram, model ve o modelin temsilini sağlayan yazılım arasındaki sınırlar da her zaman net değildir [1–2]. Yazılımın geliştirilme süreci açısından bakılırsa da farklılıklar olabilir: lisansüstü öğrenciler ve araştırmacıların, kendi kullanımları için geliştirmeye başladıkları bir yazılım, ortak bir alanda etkileşim kuran küçük grubun dışında yaygın kullanıma tabi olduğunda kullanılabilirlik ve sürdürülebilirlik açısından sorunlar yaşanabilir. Bu ve benzeri yazılım mühendisliği pratikleri ve bilimsel yazılım geliştirme etkileşimleri konusunda, özellikle son yıllarda, karşılaşılan zorluklar ve kullanılan yöntemleri inceleyen artan sayıda çalışma yayınlanmıştır [3–6]. Ancak bu çalışmalarda ortaya çıkarılan

boşluk ve pratiklerin, yazılım mühendisliği güncel beceri ve yetkinlikleri envanterleri açısından standardizasyonuna olanak sağlayacak bir eşleştirmesi yapılmamıştır. Bu çalışmanın amacı, temelde SWECOM [7] kullanarak bir eşleştirme envanteri geliştirmektir. Eşleştirme, önce makale yazarının iki farklı listesinin madde madde üzerinden giderek bir temel eşleştirme yapması; daha sonra iki ayrı uzman görüşü alınarak geçerlik kontrolü ve gerekli değerlendirilen revizyonların yapılması şeklinde iki aşamalı olarak gerçekleştirilmiştir. Böyle bir eşleştirme, bilimsel yazılım geliştirme alanında çalışan ancak yazılım mühendisliği temeli olmayan araştırmacılara yönelik kısa veya uzun süreli eğitim müfredatlarının geliştirilmesi açısından anlamlı olacaktır.

Yazılım mühendisliği pratikleri ile bilimsel yazılım geliştirme alanında kullanılan yöntemler arasındaki "boşluk" yayınlarda temelde üç farklı açıdan yer almıştır denilebilir [3–6]. Birincisi, bu bölümde verilen örneklerde de bahsedildiği gibi, yazılım geliştirmenin doğasının, bir araştırma problemi için yazılım geliştirmekle, iş sektörü için yazılım geliştirme süreçlerindeki farkından kaynaklanmaktadır. İlk bölümde dile getirilenlere ek olarak, hızlı değişen ve küçük takımlarla yazılım geliştirme ve bu koşullarda proje yönetimi [8], bakımı yapılabilirlik ve taşınırlık gibi istenebilecek özelliklerin araştırma bütçesi kısıtları ve diğer gereksinimlerle çelişir olması [9] gibi bulgular bu kategoride sayılabilir.

Bir diğer "boşluk" sebebi, bilimsel yazılım geliştiricilerin ağırlıklı olarak başka disiplinlerden gelmeleri, programlama geçmişleri olmasına rağmen yazılım mühendisliği tekniklerine hakimiyetlerinin daha az olmasıdır. Bilgisayar Bilimleri ile ilgili eğitim görmüş olanlar, örneğin, sürüm kontrol ve bağımlılık belgeleme tekniklerini diğer bilimsel yazılım geliştiricilere göre daha sık kullanmaktadırlar [10]. Gereksinim yönetimi, güncel sınama yöntemleri ve tasarım modelleme gibi konuların ise önemli bulunsalar dahi belgeleme ve sürüm yönetimine göre az bilinen pratikler olduğu belirlenmiştir [13]. Ayrıca belgeleme gibi görece sık uygulanan pratiklerin bile gereken yetkinlik düzeyinde uygulanmadığı ve dolayısıyla gerekli işlevi yerine getiremediği de örnek bilimsel yazılım incelemeleriyle belirlenmiştir [12].

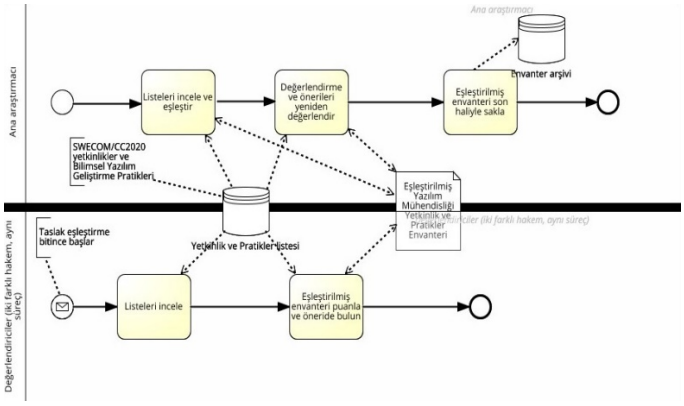
Yukardaki bulguyla ilgili bir diğer nokta da bilimsel yazılım geliştiricilerin, sınama gibi bazı alanları önemli bulsalar da yeterince bilgi sahibi olmamaları veya uygulamada eksik kalmalarıdır. Ayrıca, geliştiriciler, çevik yazılım geliştirme gibi araştırma süreçlerinin doğasında örtük olan süreçleri uyguladıklarının farkında olmayabilir. Dolayısıyla, bu konularda yazılım mühendisliği bilgi birikiminin olası katkılarından yoksun kalmaktadırlar [4,13]. Yazılım mühendisliği ile ilgili (kendi kendine öğrenme veya takım arkadaşlarından öğrenmeye ek olarak) kurumsal veya yapısal eğitimler almak ve büyük kullanıcı gruplarıyla veya büyük geliştirici takımlarıyla yazılım üretmek, yazılım mühendisliği yöntemleri farkındalığını uygulamaya geçirmede başarı etkenleri olarak değerlendirilmiştir [5,8].

Bilimsel yazılım geliştirmede yazılım mühendisliği yöntemleri eğitimi ve topluluk desteğini yaygınlaştırmaya yönelik bazı girişimler başarıyla sürdürülmektedir. Örneğin, Yazılım Marangozluğu [14] ve Kod Rafinerisi [15] girişimleri, gönüllülük ve açık kaynak esaslarına göre bilimsel yazılım geliştirmede süreç ve araç kullanımını iyileştirmeye yönelik eğitimler ve çalıştaylar düzenlemektedirler. Ancak yayınlarda temel alınan yazılım mühendisliği yetkinlik ve bilgi birikimi raporlarıyla, bilimsel yazılım geliştirmedeki gereksinimler, eksiklikler ve zorluklar üzerine yapılan çalışmaların bulguları üzerine bir eşleştirme yapılmamıştır; dolayısıyla mevcut girişimler de yazılım mühendisliğinin tüm yetkinlik birikimini kapsamamaktadırlar. Böyle bir çalışmanın gerçekleştirilmesi, bu makalenin ana motivasyonudur.

Makalenin kalan kısmı, aşağıda açıklandığı şekilde düzenlenmiştir: Bölüm 2'de bilimsel yazılım geliştirme gereksinimleri ve yazılım mühendisliği yetkinlik seçim yöntemi ve eşleştirme süreçleri aktarılmış; Bölüm 3'te bu eşleştirmeden doğan bulgu ve yorumlar, çalışmanın olası geçerlik tehditlerine karşı yapılanlar ve takip eden bir çalışmada yapılabilecekler beraber ele alınmıştır. Sonuç bölümünde ise literatürle karşılaştırılarak çalışmanın özgün değeri özetlenmiştir.

2. Yöntem

Bu çalışmada önce yazılım mühendisliği yetkinlik envanterleri araştırılmıştır. Daha sonra, güncel ve kapsayıcı bir liste çıkarılmaya çalışılmıştır. Çalışmanın bu kısmı Bölüm 2.1'de aktarılmaktadır. Sonrasında bilimsel yazılım geliştirmede yazılım mühendisliği pratikleri konusunda çalışmanın yapıldığı zaman itibariyle güncel bir sistematik haritalama makalesi olan Arvanitou ve arkadaşlarının [3] çalışması esas alınıp, bilimsel yazılım geliştirmede yazılım mühendisliği pratikleri, iyi uygulamaları, eksik ve zorluklarına yönelik bir çekirdek liste çıkarılmıştır. Arvanitou ve arkadaşları çalışmasında taranan makaleler son olarak 2019 yılındandır. Daha yeni tarihli olup aynı çalışmanın diğer tarama kriterlerine uyan makalelerle liste genişletilerek bilimsel yazılım geliştirme yazılım mühendisliği yetkinlik veya pratik gereksinim listesi oluşturulmuştur. Gereksinim listesi oluşturulması ve yetkinlik envanteriyle, yetkinlik veya pratik gereksinim listesinin eşleştirme çalışması Bölüm 2.2'de verilmiştir. Yapılan eşleştirmenin Yazılım Mühendisliği alanında çalışan iki akademisyen kodlayıcı tarafından değerlendirilmesinden ortaya çıkarılan geçerlik çalışması ise Bölüm 2.3'te aktarılmaktadır. Bu iki sürecin nasıl ilerlediği görsel olarak da bir iş süreçleri şemasıyla Şekil-1'de gösterilmiştir. Tüm çalışmanın ham verilerine (yazılım mühendisliği yetkinlikler envanteri; bilimsel yazılım geliştirme gereksinimleri listesi, eşleştirme tabloları ve hakem değerlendirmelerinin tümüne) çevrimiçi olarak ulaşılabilir [16].



Şekil-1: Yöntem akış şeması

2.1 Yazılım Mühendisliği Yetkinlikleri

Yazılım mühendisliği organizasyonlarının (özellikle, Association for Computing Machinery (ACM) ve Institute of Electrical and Electronics Engineers (IEEE) Computer Society (IEEE-CS)) üniversite ve endüstri temsilcileriyle işbirliği içinde oluşturdukları görev güçlerince üretilen birkaç yazılım mühendisliği bilgi dağarcığı ve yetkinlik raporu vardır. Bilişim Alanları Müfredatı (The Computing Curriculum 2020- CC2020) bilişim alanlarında lisans düzeyinde müfredatlar için geliştirilmiş yetkinlik bazlı bir rehberdir ve yazılım mühendisliği alanında da taslak halinde bir alt rehber içermektedir [17]. CC2020 rehberinde yetkinlik cümleleri, bilgi bileşenleri ile beceri düzeylerini eşleştirmekte; teknik becerilere inisiyatif almak gibi sosyol duygusal beceriler de eşlik etmektedir. SWECOM ise yazılım mühendisliği tabanlı iş tanımlarının farklı düzeyleri için beceri setleri ve ilgili aktiviteleri formule eden bir yetkinlik modelidir [7]. Yazılım Mühendisliği Bilgi Dağarcığı Rehberi (Software Engineering Body of Knowledge - SWEBOK) ise yazılım mühendisliği bilgi birikimini hiyerarşik bir taksonomi olarak sınıflandırmaktadır [18].

CC2020 rehberi en yakın tarihli rehber olduğu ve çalışmanın yapıldığı tarihte SWEBOK ve SWECOM'un yeni güncellemeleri üzerinde çalışılmakta olmasına rağmen kamuya açılmış güncellemeleri olmamasından dolayı yazılım mühendisliği yetkinlik envanteri çalışması için ilk aday olarak incelenmiş; ancak genelleştirilmiş yetkinlik değil, örnek olay tabanlı olması ve konfigurasyon yönetimi gibi birçok alanda yazılım mühendisliği taslak raporunun boşluklar içermesinden dolayı ana kaynak olarak tercih edilmemiştir. SWEBOK hiyerarşisi ise yetkinlikten çok statik bilgi tabanlı olduğu için bilimsel yazılım geliştirme yayınlarında belirlenen pratiklerle eşleştirilmesinin daha zor olacağı değerlendirilmiştir. Dolayısıyla yazılım mühendisliği yetkinlik envanteri kapsamında tüm SWECOM yetkinlikleri ve buna ek olarak SWECOM'da karşılığı olmadığı için CC2020'den proje yönetimi ve bireysel ve takım çalışması sosyal yetkinlikleriyle ilgili davranışsal öznitelik başlıkları altında olan yetkinlikler kullanılmıştır. Envanter, üç düzeyli kırılımla ana yetkinlik-yetkinlik-alt yetkinlik şeklinde bir organizasyonla toplam 254 alt düzey yetkinlikten oluşmaktadır ve çalışmanın veri tablolarında İngilizce asıl ifadeleriyle mevcuttur [16].

2.2 Bilimsel Yazılım Geliştirme Gereksinimlerinin Oluşturulması ve Eşleştirilmesi

Bilimsel Yazılım Geliştirme yetkinlikleri gereksinim listesinin oluşturulması için iki aşamalı bir yöntem kullanılmıştır. İlk olarak Arvanitou ve arkadaşlarının [3] bilimsel yazılım geliştirme süreçlerinde yer aldığını veya eksikliğini belirlediği, sistematik taramayla belirlenen araştırmalardan çıkarılan yazılım geliştirme pratikleri ve yetkinlikleri kataloğu, gereksinim listesinin temeli olarak alınmıştır. Arvanitou ve arkadaşları çalışmalarında sistematik bir şekilde IEEE Explore, ACM, Science Direct ve Springer sayısal kütüphanelerinde yer alan ve 2019 yılının sonuna kadar olan sürede yayınlanmış olan 359 ana çalışmayı kullanmışlardır. Arvenitou ve arkadaşları bu ana çalışmalarda toplamda en az 9 defa (en fazlası 41 defa) yer alan pratikleri yeterince sık kabul edip tablolaştırarak bilimsel yazılım geliştirmeyle en ilgili 20 yazılım mühendisliği pratiğinin listesini oluşturmuşlardır (bkz [3]'te, bu çalışmanın veri tablolarında da tekrarlanan Çizelge-10). Arvanitou ve arkadaşlarının listesinde olmayan daha yeni veya farklı dijital dizinlerde olup diğer ölçütleri sağlayan bazı ana makalelerde [10–11,19] belirtilen gereksinimler de eğer Arvanitou ve ark. listesinde bir maddeyle birebir eşleşmiyor veya daha üst bir kategori tarafından içerilmiyorlarsa, listeye eklenmiştir; araştırmacı-geliştiricinin "yeterince bireysel zamanı olmaması" gibi genel proje yönetimiyle ilgili olabilecek ancak yazılım mühendisliği veya yazılım proje yönetimiyle ilişkilendirilmesi doğrudan mümkün olamayan genel bulgular kapsam dışı tutulmuştur. Böylece Çizelge-1'de alıntı yapılan araştırmalar bazında alfabetik sırayla görülebileceği gibi toplam 32 maddeden oluşan bilimsel yazılım geliştirirken özellikle yazılım mühendisliği kapsamında ihtiyaç duyulan bir yazılım geliştirme pratikleri gereksinim listesi oluşturulmuştur. Listenin orijinal İngilizce dökümlerine çalışmanın veri tablolarından ulaşılabilir [16].

Yazılım mühendisliği yetkinlikleri envanteriyle, bilimsel yazılım geliştirme süreçleri yetkinlik gereksinim listesi eşleştirmesi bu paragrafta aktarılacak olan yöntemle yapılmıştır. Her bilimsel yazılım geliştirme yetkinlik gereksinimi (yani 32 maddeden oluşan listedeki her bir madde) için, SWECOM-CC2020 kaynaklı yazılım mühendisliği yetkinlikleri envanterinin tüm maddeleri taranmış; en ilgili en fazla üç madde olmak üzere (daha az maddeyle eşleşiyorsa en az bir, en fazla üç olmak üzere eşleştiği kadar) kapsamı en çok örtüşen maddeler Çizelge-2'deki Türkçeleştirilmiş örneklerde görüldüğü üzere eşleştirme tablosuna eklenmiştir. Yazılım mühendisliği yetkinlik envanteri (SWECOM) üç düzeyli hiyerarşik bir organizasyona sahiptir; dolayısıyla eşleştirmelerin detay düzeyinin yüksek olması için örtüşme düzeyinin en çok olduğu mümkün olan en alt, yani en detaylı kırılım kullanılmıştır. İstisna olarak, eğer ikiden fazla en alt düzey kırılım aynı anda bir bilimsel yazılım gereksinimi yetkinliği ile eşleşiyorsa, bir üst düzey başlık üç eşleştirme hanesinden birine yazılmıştır. Bu durumda daha üst düzey bir yetkinliğin tümüne gereksinim duyulduğu varsayılmıştır. Çizelge-2'de örnekleri olan eşleştirmelerin tümünün orijinal İngilizce dökümüne yine çalışmanın veri tablolarından ulaşılabilir [16].

Çizelge-1: Bilimsel Yazılım Geliştirme Süreçlerinde Yazılım Mühendisliği Pratikleri Gereksinimleri

Pratik Adı	Kaynak	Pratik Adı	Kaynak
Alan Yönelik Diller	[3]	Tasarım ve Mimari Modelleme	[3]
Biçimsel Test Yöntemleri		Yazılım Mimarileri	
Bileşen-Tabanlı Yazılım Geliştirme		Yazılım sınama (bağlanımlı, otomatik, kahinli/kahinsiz)	
Birleştirilmiş Geliştirme Ortamları (IDE)		Yazılım Süreç İyileştirme ve Döngü Yönetimi	
Çevik Yöntemler		Bağımlılıklar Hakkında Bildirim	[10]
Geliştirme Çerçevesi Önerme		Sürüm Kontrol Sistemleri Kullanımı	
Gereksinim Belirtimi		Veri Yönetimi	[11]
İnsan Yönetimi ve İletişim		Yazılım Gereksinimleri Yönetimi	
Kalite Optimizasyonu		Yazılım Performansı Optimizasyonu	
Kod Kütüphaneleri ve Arayüzleri (API) Kullanımı		Yazılım Sürüm Oluşturma Mühendisliği	
Kod Üretme		Destek Araçları Kullanımı	[19]
Ölçümlü veya Nitel Kalite Yönetimi		Eleman Bulmak ve Geliştirici Devrini Yönetme	
Paralel veya Dağıtık Programlama		En İyi yöntemler ve Hızlı İş Yetiştirme Arasındaki Denge	
Programlama Dili Edinimi ve Kullanımı		Gereksinim Toplama ve Dokümantasyonu Araçları Kullanımı	
Programlama Teknikleri		Modern Yazılım Mühendisliği Araçları Kullanabilme	
Proje Yönetimi		Yazılım Mimarisi ve Tasarım Araç Kullanımı	

Çizelge-2: Bilimsel Yazılım Geliştirme Gereksinimleri ile SWECOM/CC2020 Yetkinlikleri Eşleştirme Örnekleri

Gereksinimler	Eşleştirilen Yetkinlikler
Kod Kütüphaneleri ve Arayüzleri (API) Kullanımı	Yazılım Yapımı Planlama (SCP)*
Eleman Bulmak ve Geliştirici Devrini Yönetme	Proje Yönetimi (PM)
	Davranışsal Yetkinlikler (BA)
Veri Yönetimi	Yazılım Tasarımı Temel Yetkinlikleri (SDF)
	Yazılım Konfigurasyon Yönetimi Planlaması (PSCM)
	Yazılım Kalitesi İstatistiksel Kontrol (SQSC)

*Parantez içindeki kodlar orijinal SWECOM/CC2020 yetkinliklerini veri setinde kodlamak için kullanılmıştır.

2.3 Eşleştirme Geçerlik Kontrolü

Çalışmanın yazılım mühendisliği uzmanlığı olan tek bir araştırmacı tarafından yapılmış olması geçerlik çalışması ihtiyacını doğurmaktadır. Geçerlik kontrolü için uzmanlığı yazılım mühendisliği olan iki değerlendirici veya kodlayıcı, yapılan eşleştirme çalışmasını, 32 maddeden her biri için 1 le 3 üzerinden puanlandırmışlardır. Değerlendirici veya kodlayıcılar, puanlamaları birbirlerinden bağımsız ve birbirleriyle iletişim kurmadan yapmış, aradaki farklılıklar daha sonra araştırmacı tarafından ele alınmıştır. Yapılan eşleştirmenin uygunluğuna dair (bir pratik birden fazla yetkinlikle eşleştirilmişse hepsini toplu değerlendirerek) aşağıdaki değerlere göre bir kodlama yapmışlardır: 1-Uygun değil; 2-Nötr (Emin olamama hali dahil) 3-Uygun. Üçlü basit bir skala seçimi ile değerlendiricilerin tereddütte kalmadan araştırmacının yaptığı eşleştirmeleri yorumlayabilmeleri hedeflenmiştir. Değerlendirici veya kodlayıcının, "Uygun" kararını, araştırmacı tarafından yapılan eşleştirmenin, değerlendiricinin kişisel yazılım mühendisliği bilgisi ve SWECOM/CC2020 tabanlı yazılım mühendisliği yetkinlik envanterinin ilgili maddeleriyle uyumlu olduğu değerlendirmesi üzerine vermesi istenmiş; her değerlendirici veya kodlayıcıya birebir bilgi ve örnek aktarımı yapılmıştır. Ayrıca kodlayıcılar, uygun bulmadıkları ve emin olamadıkları eşleştirmeler için isteğe bağlı bir yorum sütununa, yorum ve önerilerini girmişlerdir. Yazılım mühendisliği yetkinlik envanteri listesini kolay taramaları ve ayrıntılı referanslara bakabilmeleri için gerekli yönergeler kendilerine verilmiştir [16]. Bu yorumların değerlendirmesi Bölüm 3.1'de yapılmıştır ve sonucunda mevcut çalışmanın yorumlara göre revize hali de ayrı bir tabloyla oluşturulmuştur. İki kollayıcının her madde için yaptığı değerlendirmeler ham haliyle [16] da verilen veri tablosunun orijinal pratik yetkinlik eşleştirme yaprağında görülebilir.

İki uzman kodlayıcının, yanıt özet değerleri ve kodlayıcılar arasındaki uyum Çizelge-3'de görülebilir. İki kodlayıcı arasında uyum %69 olarak gerçekleşmiş ve orta düzey olarak kabul edilen şansa göre düzenlenmiş kodlayıcı uyum katsayısı 0,22 (Cohen Kappa değeri) ve 0,61 (Gwet AC1 değeri) olarak hesaplanmıştır [20–21]. İki kodlayıcının da uygun bulduğu (yani 3 ile kodladığı) eşleştirme sayısı 32 üzerinden 20; ve birinin uygun (3 ile kodladığı), diğerinin nötr değerlendirdiği (2 ile kodladığı) eşleştirme sayısı 8; ikisinin de nötr değerlendirdiği (2 ile kodladığı) eşleştirme sayısı 2; ve bir kodlayıcının nötr (2 ile kodladığı), diğer kodlayıcının uygun değil şeklinde değerlendirdiği (1 ile kodladığı) eşleştirme sayısı 2'dir. Bu veriler çalışmanın geçerliği açısından yeterli olarak değerlendirilmiştir.

Çizelge-3: Ek kodlayıcı Değerlendirmeleri

Kodlayıcı_1		Kodlayıcı 2		Ortak Kodlar	
Değer*	Sıklık	Değer	Sıklık	Değer	Sıklık
3	26	3	22	İkisi de 3	20
2	4	2	10	İkisi de 2	2
1	2	1	0	Biri 3, biri 2	8
	32		32	Biri 2, biri 1	2

* 1-Uygun değil; 2-Nötr (Emin olamama hali dahil) 3-Uygun.

3. Bulgu ve Yorumlar

Bu bölümde eşleştirme çalışması sonucu bilimsel yazılım geliştirme gereksinimleri hakkında elde edilen ana bulgular, geçerlik kontrolü ile yapılan genişletmeler ve bulguların doğurduğu sonuçlar ele alınarak, geçerlik tehditleri açısından mevcut çalışmanın risklerinin nasıl azaltıldığı ve bu tehditlere bağlayarak gelecekte daha detaylı nasıl bir çalışma yapılabileceği aktarılmaktadır.

3.1 Bulgular

Bölüm 2.2'de yöntemsel olarak anlatıldığı ve Çizelge-2'de örneklendiği üzere yapılan eşleştirme çalışmasında 32 maddeden oluşan bilimsel yazılım geliştirme yetkinlik gereksinim listesine, SWECOM'un tümü ve CC2020 den yapılan eklerle oluşturulan 254 maddelik yetkinlik envanterinden toplam 56 eşleştirme yapılmıştır. Bu eşleştirmeler 32 bilimsel yazılım geliştirme gereksiniminden 5'i için maksimum eşleştirilebilecek yetkinlik sayısı olan 3'er madde; 14 gereksinim için 2'şer madde ve 13 gereksinim için birer madde olarak gerçekleşmiştir. Üç ve üstü sıklıkla eşleştirilen yazılım mühendisliği yetkinlik kategorilerinin yer aldığı Çizelge-4'te görüleceği üzere en çok eşleştirme gerçekleştirilen (toplam 13 eşleştirme) yetkinlikler, Yazılım Yapımı alanındadır. Özellikle Yazılım Yapımı Planlama (6 eşleştirme), Ayrıntılı Tasarım ve Kodlama (3 eşleştirme) en çok ihtiyaç duyulan alt yetkinlikler olarak ortaya çıkmakta ve alt kırılımlara girmeyip en üst kırılımında Yazılım Yapımı (4 eşleştirme) altında yapılan ek eşleştirmeler ile toplamda bu alanda 13 eşleştirme yer almaktadır. İkinci sırada ise toplamda 10 eşleştirme ile Yazılım Tasarımı üst yetkinliğinin ikinci düzey kırılımları olan Yazılım Tasarımı Temelleri (3 eşleştirme), Yazılım Mimarisi Tasarımı (3 eşleştirme), Yazılım Tasarımı Stratejileri ve Yöntemleri (1 eşleştirme) ve Yazılım Tasarımı Kalite Analizi ve Değerlendirmesi (1 eşleştirme), ve alt kırılımlara bölünemeyen iki ek Yazılım Tasarımı üst yetkinliğiyle beraber yer almaktadır. Bu yetkinlikleri, Yazılım Konfigurasyon Yönetimi (5 eşleştirme) ve onu takiben Proje Yönetimi (4 eşleştirme) ve Yazılım Kalite Aktiviteleri (4 eşleştirme) izlemektedir. Üçer eşleştirme sıklığı ile gereksinim duyulan diğer yetkinlikler olarak ise Yazılım Gereksinim Yetkinlikleri, Yazılım Süreç ve Yaşam Döngüsü Aktiviteleri ve Yazılım Sınama Aktiviteleri yer almaktadır. Kalan 12 eşleştirme diğer yetkinlik başlıkları altında dağılmıştır. Eşleştirmelerin tümünün sıklık-sıralı orijinal İngilizce dökümüne yine çalışmanın veri tablolarından ulaşılabilir.

Bu sıklık analizinden çıkarılabilecek bir yorum, literatürde bilimsel yazılım geliştirme alanında yapılan çalışmalarda en çok ihtiyaç duyulan yetkinliklerin yazılım tasarımı temel teknikleri (soyutlama, koşut zamanlılık, yeniden yapılandırma vb); yazılım mimarisi seçimi ve uygulaması ve tasarım kalitesinin ve stratejisinin yönetilmesi üzerinde olduğudur. Tasarım aşamasıyla beraber yazılım yaşam döngüsünde sadece klasik anlamda kodlama değil; probleme uygun çerçeve, platform, programlama dilleri, kütüphaneleri ve geliştirme ortamlarının seçimi, tasarım örüntüleri kullanımı, güvenli kodlama, kot standartlarının geliştirilmesi, kod generasyonu gibi öğeleri de içeren detaylı tasarım tekniklerinin yetkinliklerinin de öne çıktığı görülmektedir. Bu iki başlık, konfigürasyon ve proje yönetimi teknikleriyle beraber bilimsel yazılım geliştiricilerin yazılım mühendisliği bilgi ve becerileri açısından en ihtiyaç duydukları yetkinlikler olarak ortaya çıkmıştır. Sınama, konfigürasyon yönetimi becerilerine olan gereksinimler daha çok dile getirilmiş olsa bile tasarım ve yapım aşamasındaki modern yazılım mühendisliği tekniklerine duyulan gereksinim, mevcut literatürde bu şekilde bir envanter eşleştirilmesi yapılmış olmadığı için yeterince belirgin değildir [4,11]; bu sıklık çalışmasının getirdiği öncelik mevcut çalışmanın özgün bir katkısıdır.

Çizelge-4: Bilimsel Yazılım Geliştirme Yetkinlikler Envanteri Sıklık Çizelgesi

Yetkinlikler	Eşleştirme Sıklığı
Yazılım Yapımı (SC)*	13
Yazılım Yapımı Planlama (SCP)	6
Detaylı Tasarım ve Kodlama	3
Yazılım Yapımı -Ayrıştırılmamış (SC)	4
Yazılım Tasarımı (SD)	10
Yazılım Tasarımı Temelleri (SDF)	3
Yazılım Mimarisi Tasarımı (SAD)	3
Yazılım Tasarımı Stratejileri ve Yöntemleri (SDSM)	1
Yazılım Tasarımı Kalite Analizi ve Değerlendirmesi (SDQAE)	1
Yazılım Tasarımı-Ayrıştırılmamış (SD)	2
Yazılım Konfigurasyon Yönetimi (SCM)	5
Proje Yönetimi (PM)	4
Yazılım Kalite Aktiviteleri (SQ)	4
Yazılım Gereksinim Yetkinlikleri (SR)	3
Yazılım Süreç ve Yaşam Döngüsü Aktiviteleri (SPLC)	3
Yazılım Sınama Aktiviteleri (ST)	3
*Sadece üst kırılım bazında üçten fazla sıklıkla tekrarlanan yetkinlikler verilmiştir. Tüm liste veri setinde mevcuttur. Parantez içindeki kodlar orijinal SWECOM/CC2020 yetkinliklerini veri setinde kodlamak için kullanılmıştır.	

Geçerlik kontrolü çalışması sonucu kodlayıcı uzmanlardan iki türlü öneri gelmiştir: Mevcut 5 eşleştirmenin envanterde kalıp kalmaması konusunda emin olamadıklarını belirten "çıkarma" önerileri ve ek olarak mevcut envanterde olmayan 2'si en alt kırılım düzeyinde alt yetkinlik, 4'ü bir üst kırılım düzeyinde yetkinlik olmak üzere 6 "ekleme" önerisi. İki listenin de arka planını oluşturan referanslardan yapılan kontroller sonucu, çıkarılması açısından tereddüt oluşturan öğelerin, eşleştirme açısından doğru olduğu teyit edilmiş ve çıkarma yapılmamıştır [3,7]. Mevcut envanterde olmayan ekleme önerileri ise aynı şekilde kontrolden sonra uygun bulunarak eklenmiştir ve çalışmanın revize veri tablolarından ulaşılabilir. Yapılan eklemeler, bu bölümde yer alan sıklık yorumlarında bir farklılık yaratmamıştır.

3.2 Geçerlik Tehditleri ve Yapılabilecek Çalışmalar

Yapılan çalışma, büyük ölçüde, Arvenitou ve arkadaşlarının [3] sistematik tarama çalışmasındaki bulgulara dayanmaktadır. Arvanitou ve arkadaşları çalışmalarını aslında yazılım kalitesi bileşenleri ile yetkinlik ve süreç gereksinimlerini eşleştirme amaçlı gerçekleştirmiştir. Yapılan taramanın sistematiği sağlam olmakla beraber geçerlik endişeleri açısından birinci

elden yani bilimsel yazılım geliştiriciler topluluğundan doğrudan veri elde etmenin, daha somut ve sağlam bir envanter çıkarabilmek açısından yararı olacaktır. SWECOM, SWEBOK ve CC2020 gibi yetkinlik/bilgi envanterlerinin üzerinde çalışılan yeni sürümleri kamuyla paylaşıldıktan sonra, bu envanterler üzerinden detaylı bir anket çalışmasıyla bilimsel amaçlı yazılım geliştiren araştırmacılara uluslararası düzeyde ulaşılıp daha detaylı ve birinci elden veri toplayarak geçerliği yüksek bir tekrar çalışması yürütülebilir.

Benzer şekilde, SWECOM [7] yetkinlik envanterinde, yetkinlikler, teknisyenden, deneyimli yazılım mühendisine kadar değişken bir yetkinlik yelpazesini gösteren beşli bir ölçekte iş tanımlarının farklı aktivitelere göre eşleştirmesiyle ("Takip eder"; "Yardım eder"; "Katılır"; "Liderlik eder" gibi) detaylandırılmışlardır. Mevcut çalışmada bu detaya inebilmek mümkün olmamıştır; çünkü bilimsel yazılım geliştirme süreçlerindeki gereksinimlerin farklı iş tanımları ve aktivitelere göre detaylandırıldığı bir çalışma henüz literatürde görülmemektedir. Yukarıdaki paragrafta önerilen çalışma, bu tip bir çeşitlemeye de zemin sağlayabilir.

Eşleştirme çalışmasının tek uzmanla yürütülmesinin sakıncaları, Bölüm 2.3'te aktarıldığı gibi, iki farklı uzmanca yapılan kodlamalarla sağlama ve yapılan önerileri kontrol edip, envantere katarak ulaşılan genişletme çalışmasıyla hafifletilmiştir. Birinci elden elde edilen veriyle yapılacak bir çalışmada, eş güdümlü çalışan bir grup uzmanın eş zamanlı, istatistiki kontrolle eşleştirme yapması çalışmanın iç geçerliğine katkı sağlayacaktır.

4. Sonuç

Bu çalışmada, bilimsel yazılım geliştirme sürecindeki yetkinlik gereksinimleri, yazılım mühendisliği yetkinlik envanterleriyle (SWECOM-CC2020) eşleştirilerek, hem ihtiyaç duyulan yetkinliklerin standart terminolojiyle kullanılması, hem de farklı bilim dallarından bilimsel yazılım geliştiricilerin hangi yazılım mühendisliği pratiklerini öğrenmeye ve uygulamaya en çok ihtiyaçları olduğunun alan rehberleri yetkinliklerine odaklanarak belirlenmesi hedeflenmiştir. Yayın özetinde belirtildiği gibi çeşitli çalışmalarda anket gibi yöntemlerle elde edilmiş, bilimsel yazılımların geliştirilmesinde gereksinim duyulan yetkinlik listeleri mevcuttur [3–5,19]. Bu çalışmanın, alan yayınlarda bahsedilen çalışmalara ek olarak getirdiği katkılar:

1. Kapsamlı bir yetkinlik gereksinim listesi oluşturulması,
2. IEEE ve ACM'in yazılım mühendisliği ölçünlü yetkinlik envanterleriyle bir eşleştirme sağlayarak, terminolojinin birleştirilmesi,
3. Yetkinlik envanterlerinin tümü ele alındığında hangi alanların kapsandığının veya kapsanmadığının daha iyi değerlendirilebilmesi olmuştur.

Bu çalışmanın ışığında, özellikle tasarım ve yapım aşamasında bilimsel yazılım geliştirme için gerekli yetkinliklerin mevcut bilimsel yazılım geliştirme eğitimleri girişimlerinde yeterli düzeyde kapsanmadığı gözlenmiştir [14–15]. Ortaya çıkan eşleştirme listesi, Yazılım Marangozluğu [14] ve Kod Rafinerisi [15] gibi mevcut uluslararası girişimlerin müfredatlarının yenilenmesine temel oluşturabileceği gibi; ulusal düzeyde de Ar-Ge politikaları geliştirirken, üniversite araştırma merkezleri ve yazılım mühendisliği dışındaki bilim alanlarından lisansüstü eğitim öğrencilerinin ve diğer araştırmacıların ulaşabileceği yazılım mühendisliği teknikleri eğitim programları tasarlanmasında da kullanılabilir.

Teşekkür

Atılım Üniversitesi Yazılım Mühendisliği Bölümü öğretim üyeleri Prof. Dr. Ali Yazıcı, Doç. Dr. Çiğdem Turhan ve Bilişim Sistemleri Mühendisliği Bölümü öğretim üyesi Dr. Öğr. Üyesi Tuna Hacaloğlu'na çalışmaya yaptıkları katkı için teşekkür ederim.

Kaynakça

[1] Johanson, A., & Hasselbring, W. (2018). Software Engineering for Computational Science: Past, Present, Future. *Computing in Science and Engineering*.doi:10.1109/MCSE.2018.108162940

[2] Taatgen, N. A., Vugt, M. K. van, Borst, J. P., & Mehlhorn, K. (2016). Cognitive modeling at ICCM: state of the art and future directions. *Topics in Cognitive Science*. *8*(1), 259-263. doi:10.1111/tops.12185.

[3] Arvanitou, E. M., Ampatzoglou, A., Chatzigeorgiou, A., & Carver, J. C. (2021). Software engineering practices for scientific software development: A systematic mapping study. *Journal of Systems and Software*, *172*. doi:10.1016/j.jss.2020.110848

[4] Heaton D., & Carver, J. (2015). Claims about the use of software engineering practices in science: A systematic literature review. *Information and Software Technology, 67*, 207-219. doi:10.1016/j.infsof.2015.07.011.

[5] Storer, T. (2017). Bridging the chasm: a survey of software engineering practice in scientific programming. *ACM Computing Surveys, 50*(4), 1-32. doi:10.1145/3084225.

[6] Segal, J., & Morris, C. (2008). Developing scientific software. *IEEE software*, *25*(4), 18-20. doi:10.1109/MS.2008.85

[7] IEEE. (2014). *A Software Engineering Competency Model (SWECOM)*. Version 1.0. IEEE Computer Society Press.

[8] Kurtaran, F. (2018). "An Evaluation of the Use of Software Engineering Practices by Cognitive Modeling Researchers." MSc Thesis. Department of Computer Engineering. Atilim University, Turkey. https://tez.yok.gov.tr/UlusalTezMerkezi/

[9] Wagner S., Pflüger, D., & Mehl, M. (2015). Simulation software engineering: experiences and challenges. *Proceedings of the 3rd International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering - SE-HPCCSE '15,* 1-4. doi:10.1145/2830168.2830171.

[10] AlNoamany Y, Borghi JA. (2018). Towards computational reproducibility: researcher perspectives on the use and sharing of software. *PeerJ Computer Science* 4:e163. doi:10.7717/peerj-cs.163

[11] Wiese, I., Polato I. & Pinto, G. (2020). Naming the Pain in Developing Scientific Software. *IEEE Software*, 37(4), 75-82, July-Aug. 2020, doi:10.1109/ms.2019.2899838.

[12] Hermann, S., Fehr, J. (2022). Documenting research software in engineering science. *Scientific Reports,* 12 (6567). doi:10.1038/s41598-022-10376-9

[13] Sanders, R. & Kelly, D., 2008. Dealing with Risk in Scientific Software Development. *IEEE Software*, 25(4), pp.21–28. doi:10.1109/ms.2008.84.

[14] Software Carpentry. https://software-carpentry.org/. Erişim Tarihi: Haziran, 2022.

[15] Code Refinery. https://coderefinery.org/ . Erişim Tarihi: Haziran, 2022.

[16] Say, B. Bilimsel Yazılım Geliştirme Yetkinlik Envanteri Çalışması Tabloları https://drive.google.com/drive/folders/10SP32GlOXTNXKBxstVBhme1XkfWJXwwm?usp=sharing. Erişim Tarihi: Kasım, 2022.

[17] CC2020 Task Force. (2020). *Computing Curricula 2020: Paradigms for Global Computing Education*. *Computing Curricula 2020*. ACM. doi:10.1145/3467967

[18] Bourque P. & Fairley, R. E. (Ed.) (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3.0.* Retrieved from https://www.computer.org/education/bodies-of-knowledge/software-engineering

[19] Carver JC, Weber N, Ram K, Gesing S, Katz DS. (2022). A survey of the state of the practice for research software in the United States. *PeerJ Computer Science*, doi:10.7717/peerj-cs.963

[20] The jamovi project (2022). jamovi. (Version 2.3) [Yazılım]. https://www.jamovi.org.

[21] Balcı, S. (2022). ClinicoPath jamovi Module doi:10.5281/zenodo.3997188. [R package]. https://github.com/sbalci/ClinicoPathJamoviModule.