Yazılım Mühendisliği Eğitimi

N.Yasemin Topaloğlu

Ege Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü
35100 Bornova – İzmir
e-posta: yasemin@bornova.ege.edu.tr

Özet

Yazılımın günümüzde hızla artan önemi, tüm dünyada yazılım mühendisliği disiplinindeki çalışmaların yoğunlaşmasına neden olmuştur. Son yıllarda, IEEE ve ACM gibi uluslararası mesleki kuruluşlar tarafından yazılım mühendisliği çekirdek bilgisinin tanımlanması ve bu bilgilerle uyumlu yazılım mühendisliği eğitim programlarının geliştirilmesine yönelik çalışmalar yapılmaktadır. Bu bağlamda, diğer mühendislik dallarında olduğu gibi yazılım mühendisliği için de ayrı eğitim programlarının oluşturulması gündeme gelmiştir. Yazılım mühendisliği disiplinin olgunlaşma sürecinde yazılım mühendisliği eğitimi özel bir önem kazanmıştır. Bu bildiride, yazılım mühendisliği eğitiminin kapsamı ve özellikleri incelenmiştir.

Abstract

The increasing importance of software caused intensive studies in the discipline of software engineering in all over the world. In the last couple of years, international professional societies like IEEE and ACM initated projects which aim to define the core knowledge of software engineering and to develop software enginnering curriculum based on this knowledge. During the process of the maturation of the software engineering discipline, the software engineering education gained a special importance. In this paper, the knowledge areas that should be covered in the software engineering education and the general characteristics of the software engineering education are discussed.

1.Giriş

Yazılım Mühendisliği, 1968 yılında yazılım geliştirmedeki problemleri tartışmak ve gerekli çalışmaları tanımlayabilmek için NATO tarafından düzenlenen bir konferansta, güvenilir ve etkin olarak çalışan yazılımların ekonomik olarak elde edilebilmesi için mühendislik ilkelerinin uygulanması olarak tanımlanmıştır[1]. 1968 yılında düzenlenen bu konferanstan bugüne kadar Yazılım Mühendisliği disiplini çok gelişme kaydetmiş ve ilerlemiş olmakla beraber, günümüzde hala diğer mühendislik dalları ile eş tutulmamaktadır[2]. Bununla birlikte, yazılımın hayatımızdaki artan önemi ve ülke ekonomilerine etkileri, yazılım mühendisliğine olan ilgiyi hem akademik çevrelerde hem de endüstriyel platformlarda artırmıştır. Son yıllarda yazılım geliştirmeye önemli katkıları olan yazılım geliştirme metodolojileri, programlama paradigmaları, programlama dilleri ve araçlar geliştirilmiştir. Günümüzde yazılım geliştirme alanında milyonlarca kişi çalışmakta, yazılım mühendisliğinde yer alan çeşitli konular üzerinde çeşitli konferanslar düzenlenmekte ve yazılım mühendisliği üzerine eğitim programları bulunmaktadır. Bütün bu gelişmelere paralel olarak yazılım mühendisliğinin geçmişe oranla daha iyi tanındığını ve diğer meslek dalları arasındaki yerinin belirginleştiğini söyleyebiliriz. Ancak yazılım mühendisliği çevrelerinin üzerinde birleştiği bir konu, yazılım mühendisliğinin halen "*genç*" bir disiplin olduğu ve yeterli olgunluğa ulaşması için bir çok konuda fikir birliğinin sağlanması için zamana gereksinim olduğudur[3].

Yazılım mühendisliğinin olgunlaşma sürecini hızlandırmak için yakın geçmişte, uluslar arası meslek uruluşları olan *Association for Computing Machinery (ACM)* ve *IEEE Computer Society (IEEE CS*), "Yazılım Mühendisliği Koordinasyon Komisyonunu" oluşturmuş ve yazılım mühendisliği alanında çeşitli projelere başlamıştır[4]. Bunlar;

- Yazılım mühendisliği çekirdek bilgisinin tanımlanması - *Software Engineering Body of Knowledge (SWEBOK),*
- Yazılım mühendisliği etiklerinin tanımlanması - *Software Engineering Code of Ethics and Professional Practice (SWCEPP),*
- SWEBOK ile uyumlu olarak örnek bir eğitim programı tanımlanması *Software Engineering Education Project(SWEEP),*

olarak belirlenmiştir. Yazılım mühendisliği çekirdek bilgisinin(SWEBOK) ilk versiyonu, çalışmaların ilk aşamasının tamamlanmasıyla, 2001 yılında hazırlanmıştır. SWEEP kapsamında eğitim programlarının tanımlanması çalılşmaları halen devam etmektedir. Yukarıda belirtilen çalışmalarla bağlantılı olarak, son yıllarda A.B.D, Avustralya ve Kanada'da çeşitli üniversiteler "Yazılım Mühendisliği Lisans Programları" oluşturmuşlardır[2].

Günümüzde yazılımlarda karşılaşılan problemler, yazılım mühendisliği eğitiminin önemini vurgulamaktadır. Bu bağlamda, yazılım problemlerinin çözümü ve iyileştirmeler için, gelecekteki yazılım mühendislerinin eğitimine odaklanılması gerektiği, literatürde sıklıkla vurgulanmaktadır[5] [6].

Bu bildiride, yazılım mühendisliği eğitiminin önemli bileşenleri incelenecek ve literatürde bu konuda yapılan çalışmalar tanıtılacaktır. Bu kapsamda, bildirinin ikinci bölümünde, IEEE ve ACM tarafından kurulan komisyonun tanımladığı *Yazılım Mühendisliği Eğitimi Bilgisi*'nde (Software Engineering Education Knowledge) [7] yer alan bilgi alanları tanıtılmıştır. Üçüncü bölümde ise yazılım mühendisliği eğitim programlarının ana özellikleri incelenmiştir. Dördüncü bölüm, sonuç bölümünü içermektedir.

2.Yazılım Mühendisliği Eğitimi Bilgi Alanları

1998 yılında *IEEE CS* ve *ACM*, bilgisayar lisans programlarının müfredatlarına yönelik rehberlerin gözden geçirilmesi ve yenilenmesi için ortak bir kurul oluşturmuşlardır. Bilgisayar Müfredatları (*Computing Curricula*) adı verilen bu etkinlikte, Bilgisayar Bilimleri, Bilgisayar Mühendisliği, Yazılım Mühendisliği ve Bilgi Sistemleri alanlarında çalışmalar yapılmış ve her bir konu için raporlar hazırlanmıştır[8]. Kurulun yazılım mühendisliği müfredatıyla ilgili grubu, ilk olarak bir yazılım mühendisliği lisans programı için aşağıda özetlenen amaçları, bir yazılım mühendisliği lisans programının mezununun sahip olması gereken yetenekler açısından tanımlamıştır:

1. Yazılım ürünleri geliştirmek için bir takımın parçası olarak çalışmak,
2. Kullanıcı gereksinimlerini belirlemek ve onları yazılım gereksinimlerine çevirmek,
3. Çelişen amaçları düzenlemek, maliyet, zaman, bilgi ve organizasyon kısıtlamaları içinde kabul edilebilir uzlaşmalar bulmak.
4. Bir veya daha çok uygulama alanı için, etik, sosyal, yasal ve ekonomik ilgileri bütünleştiren mühendislik yaklaşımlarını kullanarak uygun çözümler tasarlamak.
5. Yazılım tasarımı, geliştirilmesi, gerçekleştirimi ve doğrulanması için bir temel sağlayan mevcut teorileri, modelleri ve teknikleri anlamak ve uygulayabilmek.
6. Tipik bir yazılım geliştirme ortamında etkin olarak çalışmak, gerekli olduğunda liderlik yapabilmek ve kullanıcılarla iyi iletişim kurabilmek.
7. Yeni modelleri, teknikleri ve teknolojileri öğrenebilmek.

Kurul, bu amaçlar doğrultusunda, on temel alandan oluşan Yazılım Mühendisliği Eğitimi Bilgi Alanları (*Software Engineering Education Knowledge Areas*) tanımlamıştır. Yazılım Mühendisliği Eğitimi Bilgi Alanlarında tanımlanan bilgi alanları, bir müfredat oluşturmamakta, sadece yazılım mühendisliği eğitim programlarının tasarlanması ve gerçekleştirilmesi için bir temel sağlamaktadır.

Yazılım Mühendisliği Eğitimi Bilgi Alanları aşağıda açıklanmıştır[7]:

1. *Temeller:* Yazılım mühendisliğinin temelleri, yazılım mühendisliğinin ürettiği ürünlerin niteliklerini anlatan teorik ve bilimsel temellerden, bu ürünleri modellemeyi ve tanımlamayı kolaylaştıran matematiksel temellerden ve öngörülebilir sonuçlar üreten ana ilkelerden oluşur. Buradaki ana nokta, kaynakları belirlenmiş bir amaca dönüştürmek için mühendislik tasarımı ve mühendislik biliminin uygulanmasıdır.

2. *Profesyonel Uygulama*: Profesyonel uygulama, yazılım mühendislerinin, yazılım mühendisliğini profesyonel ve etiğe uygun olarak uygulayabilmeleri için sahip olmaları gereken bilgi, beceri ve davranışlarla ilgilidir. Profesyonel uygulamalar bilgisi teknik iletişim, psikoloji ve sosyal ve mesleki sorumlulukları içerir.
3. *Gereksinimler:* Yazılım gereksinimleri, bir sistemin amacını ve hangi içerikte kullanılacağını tanımlar. Gereksinimler, kullanıcıların gerçek gereksinimleri ile yazılım ve diğer bilgisayar teknolojileri arasında köprü oluşturur. Gereksinimlerin belirlenmesi, sistemin fizibilite çalışmasını, kullanıcıların gereksinimlerinin analizi, sistemin ne yapacağının ve ne yapmayacağının kısıtlamalar gözönünde alınarak belirlenmesini ve bu bilginin kullanıcılar tarafından doğrulanmasından oluşur.
4. *Tasarım:* Yazılım tasarımı, bir bileşenin veya bir sistemin nasıl gerçekleştirileceğini belirlemek için kullanılan teknikler, stratejiler, gösterimler ve desenlerle ilgilidir. Tasarım, kaynaklar, performans, güvenilirlik ve güvenlik gibi kısıtlamalar gözönüne alınarak işlevsel gereksinimlere uygun olmalıdır. Ayrıca, yazılım bileşenleri arasındaki içsel arayüzler, mimari tasarım, veri tasarımı, kullanıcı arayüzü tasarımı, tasarım araçları ve tasarımın değerlendirilmesi de bu alanın kapsamındadır.
5. *Yazılım Oluşturma:* Bu alan, tasarımda belirlenmiş yazılım bileşenlerinin geliştirilmesiyle ilgili bilgileri içermektedir. Bu kapsamda, bir tasarımın bir gerçekleştirim diline çevrilmesi, bileşen sınamaları ve program belgelemeleri incelenmektedir.
6. *Yazılım Sınama ve Doğrulama:* Yazılım sınama ve doğrulama, elde edilen programın hem belirlenen gereksinimleri sağladığını hem de gerçekleştirimin beklenenlere uygun olduğunu kontrol etmek için statik ve dinamik sınama teknikleri kullanır. Statik teknikler, yazılımın tüm yaşam döngüsü boyunca elde edilen gösterimlerin analizi ve kontrolüyle ilgilenirken, dinamik teknikler sadece gerçekleştirilmiş sistemi içerir.
7. *Yazılım Gelişimi*: Yazılım gelişimi, yazılımın kullanıma verilmesinin öncesindeki ve sonrasındaki aşamalarda etkin bir maliyetle desteklenmesini sağlar. Bu destek, gelişen sistemi oluşturan versiyonların veya sürümlerin her biri için hazırlık aktivitelerine gerek duyar. Bu aktiviteler, planlama, ölçüt desteği, regresyon sınama ve karmaşıklık kontrolünü içermektedir. Bu aktiviteleri desteklemek için kullanılan teknikler, program anlama, sürüm planlaması, değişiklik tanımlaması, yeniden mühendislik, tersine mühendislik, bakım, sistemin kullanımına son verilmesini içerir.
8. *Yazılım Süreci*: Yazılım süreci, yaygın olarak kullanılan yazılım yaşam döngüsü süreç modellerinin tanımlanmasıyla ilgili bilgileri ve kurumsal süreç standartlarını; yazılım süreçlerinin tanımlanmasını, gerçekleştirilmesini, ölçülmesini, bakımını, yönetimini, değiştirilmesi ve iyileştirilmesini; ve yazılım geliştirme ve bakımı için gereken teknik ve yönetimsel aktiviteleri gerçekleştirmek için tanımlı bir süreç kullanımınını kapsamaktadır.
9. *Yazılım Kalitesi*: Yazılım kalitesi, yazılım geliştirmenin ve bakımın tümünü etkileyen ve tümünden etkilenen bir kavramdır. Hem geliştirilen ürünlerin kalitesini hem de bu ürünleri geliştirmek için kullanılan süreçlerin kalitelerini içerir. Ürün kalite nitelikleri, kullanılabilirlik, güvenilebilirlik, güvenlik, bakıma uygunluk, esneklik, etkinlik ve performans gibi kriterleri kapsamaktadır.
10. *Yazılım Yönetimi*: Yazılım Yönetimi, tüm yazılım yaşam döngüsü aşamalarının planlanması, düzenlenmesi ve izlenmesiyle ilgili bilgileri içermektedir. Yazılım geliştirme projelerinin başarısı için, farklı organizasyonel birimlerdeki işlerin koordinasyonu için, yazılım versiyonlarının bakımı için, kaynakların gerekli oldukları zaman var olabilmesi için, projedeki işlerin uygun olarak bölünebilmesi için, iletişimin kolaylaşması için kritik önemdedir.

3.Yazılım Mühendisliği Eğitim Programları

Yazılım mühendisliği eğitimi, geleneksel olarak üniversitelerin Bilgisayar Bilimleri ve Bilgisayar Mühendisliği bölümlerinde yer alan dersler ile gerçekleştirilmektedir. Bu programlarda "Yazılım Mühendisliği" adında bir ders bulunmakta, bazı programlarda ise buna ek olarak yazılım mühendisliğinin çeşitli konularını işleyen dersler yer almaktadır. Son yıllarda ise, özellikle Kanada, Avustralya ve Amerika Birleşik Devletleri'nde bulunan bazı üniversitelerde, Yazılım Mühendisliği Lisans Programları oluşturulmuştur[7]. Bununla bağlantılı olarak, yazılım mühendisliği lisans programlarının içeriği akademik çevrelerde yaygın olarak tartışılmaktadır.

Yazılım mühendisliği alanındaki önemli çalışmalarıyla tanınan David Parnas, *IEEE Software* dergisinin, yazılım mühendisliği eğitimine ayrılmış bir sayısında, Yazılım Mühendisliği eğitiminin kapsamını ve geleneksel Bilgisayar Bilimleri eğitiminden farkını incelemiştir[2]. Parnas bu makalesinde, yazılım mühendisliği hakkındaki bilgilerin artmasıyla, güvenilir yazılımların tasarlanması, gerçekleştirilmesi, sınanması ve bakımının yapılması için bu konularda bilgili mühendislere olan gereksinimin artığını belirtmiştir. Bilgisayar Bilimlerinin son 30 yılda ulaştığı olgunluk düzeyinin, yazılım mühendisliği için ayrı eğitim programları tanımlanmasına olanak sağladığı düşüncesinde olan Parnas, aynı makalede, yazılım mühendisliğinin sadece programcılık olmadığı vurgulamış ve Bilgisayar Bilimleri ve Yazılım Mühendisliği arasındaki ilişkiyi, Fizik ve Elektrik Mühendisliği arasındaki ilişkiye benzetmiştir. Bu bağlamda, mühendislerin belirli bir alandaki bilgiye ek olarak, bilgilerin uygulanması için gereki metodları öğrenmeleri gerektiğini vurgulamıştır.

Parnas[2], ayrı bir mühendislik disiplini olarak yazılım mühendisliği lisans müfredatı önermiştir. Bu müfredatta, etkin bir yazılım mühendisliği eğitimi için, yazılım mühendisliğinin bilimsel temeli (Bilgisayar Bilimleri) korunurken, geleneksel mühendislik eğitimi yaklaşımının izlenmesi gerektiği savunulmuştur. Bu bağlamda önerilen müfredatın içeriği;

- Diğer tüm mühendislik disiplinleri tarafından alınan temel dersler,
- Yazılım mühendisliğinin matematiksel temelleri üzerine dersler,
- Yazılım geliştirmeyle ilgili dersler,

olarak özetlenebilir.

Önerilen müfredatta, güncel bazı programlama dillerine ve teknolojilerine değinilmemiştir. Güncel yaklaşımların temelindeki fikirlerin öğretilmesi gerektiği ve öğrencilerin güncel araçları laboratuar çalışmalarında kullanması gerektiği ifade edilmiş, ancak, bu konuların eskilerin yerini aldığının ve zamanla onların da yerlerini yenilerine bırakacaklarının unutulmaması gerektiği belirtilmiştir. Bugünün öğrencilerinin meslek hayatlarının 40 yıl sürebileceğine dikkat çeken Parnas, o süre içinde geçerliliğini koruyacak ve kullanışlı olacak temel bilgilerin belirlenerek, derslerde vurgulanması gerektiğine dikkat çekmiştir. Benzer şekilde diğer bir makalede de, yazılım mühendisliği eğitiminin güncel dillere ve araçlara yoğunlaşmak yerine, bir mühendisin tüm meslek hayatında yararlı olacak iyi mühendislik uygulamalarına ve yazılım geliştirmenin çeşitli etkinliklerine ilişkin tekniklere yoğunlaşması gerektiği ifade edilmektedir[5].

Yazılım mühendisliği lisans programlarının geleneksel Bilgisayar Bilimleri eğitimine dayanmakla birlikte, geleneksel mühendislik programlarından alınmış bileşenler içermesi gerektiği IEEE Computer Dergisinde Mayıs 2000'de yayınlanan bir başka makalede de tartışılmıştır[3]. Bu kapsamda tanımlanan "İyi bir Yazılım Mühendisliği Programının Elemanları" nın temel bileşenleri aşağıda özetlenmiştir:

1. Yazılım ve donanım ürünleri hakkında temel teknik bilgi ve becerileri sağlayan *Bilgisayar Bilimi Temelleri.* Bunlar, programlama dilleri, modelleme, veritabanları, işletim sistemleri, ağ sistemleri, algoritmalar olarak sayılabilir.
2. Yazılımı ve ilgili belgelemeyi oluşturmak ve bakımı yapmak için gerekli teknik bilgi, yetenek ve araçları sağlayan *Yazılım Mühendisliği Temelleri*. Bunlar arasında, yazılım süreçleri, yaşam döngüsü modelleri, yazılım ölçütleri, mimari ve tasarım yöntemleri bulunmaktadır.
3. Genel sistem ilkelerinin, ekonominin ve mühendislerin görev ve sorumluluklarının anlaşılmasını sağlayan *Mühendislik Uygulamaları ve Etiği.*
4. Ekip olarak çalışmak için gerekli bilgiyi sağlayan *Etkin İletişim ve Grup Çalışması Yetenekleri.*
5. Öğrencileri gerçek hayat problemleine hazırlayan *Bir Uygulama Alanında Deneyim.*
6. Öğrencileri gereksinimlerin değişmesi, proje yönetimi, konfigürasyon yönetimi, araç kullanımı gibi konulara hazırlayan *Bir Ekip Projesi.*
7. Öğrencileri gerçek hayat ortamına hazırlayan *Endüstriyel Bir Ortamda Deneyim.*
8. Ders notu veya ders gibi belirli bir düzende sağlanmayan bilgiyi aramak, değerlendirmek ve kullanmak için gerekli beceriyi kazandırmayı amaçlayan *Yaşamboyu Öğrenme Araçları*.

Yazılım mühendisliği lisans programlarının, yazılım problemlerinin çözümüne etkisini yadsımamakla birlikte, bunun sadece bir başlangıç olduğu ve yeterli olmadığı da ifade edilmiştir[5]. Aynı çalışmada, yazılım mühendisleri için "Temel Bilginin" (*Body of Knowledge*) tanımlanmasının önemine dikkat çekilmektedir. Vurgulanan bir diğer nokta, eğitim süresince, mezunların yeni teknolojileri hızlı ve etkin bir şekilde öğrenebilmesi ve uygulayabilmesi için gerekli temellere yoğunlaşılması gerektiğidir.

Yazılım mühendisliğine yönelik eğitim programlarıyla bağlantılı olarak, öne çıkan bir diğer konu, akreditasyon konusudur. Diğer mühendislik dallarında olduğu gibi, yazılım mühendisliğinde de bir akreditasyon sisteminin oluşturulmasının hem eğitim programlarının niteliğini artırmaya hem de yazılım mühendisliği disiplininin olgunlaşmasına katkıda bulunacağı belirtilmiştir[2].

Literatürde, yazılım mühendisliği eğitimine ek olarak, yazılım mühendislerinin profesyonel meslek hayatlarının başlangıcında bir "stajyerlik" dönemini tamamlamaları da önerilmiştir[9]. Ayrıca, yazılım teknolojilerindeki hızlı değişim ve bilişim teknolojisindeki hızlı gelişmeler, mesleki eğitim programlarının da gerekliliğini ortaya çıkarmaktadır.

4.Sonuç

Yazılım mühendisliğinin olgunlaşma sürecini hızlandırmak için, yazılım mühendisliği temel bilgisinin tanımlanması ve yazılım mühendisliği temellerine odaklanmış eğitim programlarının tanımlanması çalışmaları son yıllarda hız kazanmıştır. Yazılımın hayatımızdaki yeri ve önemi göz önüne alındığında yazılım mühendisliği eğitiminin önemi daha iyi ortaya çıkmaktadır. Bu bildiride, yazılım mühendisliği eğitiminin kapsamı ve özellikleri, literatürde yer alan güncel çalışmalar ışığında incelenmiştir. Yazılım mühendisliği disiplinin gelişmesiyle birlikte, yazılım mühendisliği için diğer mühendislik dallarında olduğu gibi ayrı eğitim programlarının oluşturulmasının yakın gelecekte hız kazanacağı görülmektedir. Literatürde de belirtildiği gibi [3][5], yazılım mühendisliği disiplinin henüz genç bir alan olması nedeniyle, yazılım mühendisliği eğitiminin içeriğinin tanımlanması kapsamlı bir çalışmayı gerektirmektedir. İlgili çevrelerin katılımıyla oluşacak platformlarda konunun ele alınmasının ve tartışılmasının yararı büyüktür.

Kaynaklar

[1] Naur, P. ve Randall, B. (editörler.), "Software Engineering:A Report on a Conference Sponsored by the NATO Science Committee",NATO, 1968.

[2] Parnas, D. L. "Software Engineering Programs Are Not Computer Science Programs", *IEEE Software*, Kasım/Aralık 1999, s. 19-30.

[3] Pour,G., Griss, M. ve Lutz, M. "The Push to Make Software Engineering Respectable", *IEEE Computer*, Mayıs 2000, s. 35-43.

[4] Abran, A and Moore, J. (editörler), "Guide to the Software Engineering Body of Knowledge *SWEBOK*". *IEEE Computer Society Press*, 2001.

[5] Saiedian, H., Bagert, D., ve Mead,N., "Software Engineering Programs: Dispelling the Myths and Misconceptions", *IEEE Software*, Eylül/Ekim 2002, s. 35-41.

[6] Hilburn, T. ve Humphrey, W., "The Impending Changes in Software Education", *IEEE Software*, Eylül/Ekim 2002, s. 22-24.

[7] "Computing Curricula - Software Engineering Volume", http://sites.computer.org/ccse/

[8] "Computing Curricula", http://www.computer.org/education/cc2001/

[9] McConnell, S. ve Tripp, L. "Professional Software Engineering:Fact or Fiction?" *IEEE Software*, Kasım/Aralık 1999, s. 13-17.