Yazılım Mühendisliği Dersi için Geliştirilmiş Ders Akış Modeli ve İlgili Vaka Çalışması

Murat Yılmaz[1] ve Ulaş Güleç[1,2]

[1] Çankaya Üniversitesi, Bilgisayar Mühendisliği Bölümü, Ankara, Türkiye
{myilmaz,ulasgulec}@cankaya.edu.tr
[2] Orta Doğu Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, Ankara, Türkiye

Özet. Tüm dünyada olduğu gibi ülkemizde de çok sayıda yazılım projesi öngörülen bütçe ve süre sınırları aşılarak kullanıcı beklentileri tam olarak karşılanmadan sonlanmaktadır. Bunun önemli sebeplerinden birisi olarak yazılım mühendisliği konularına hakim iş gücü eksikliği gösterilebilir. Bu makalede, Çankaya Üniversitesi Bilgisayar Mühendisliği bölümünde son dört yıldır yürütülen CENG 396 Yazılım Mühendisliği dersi için düşünülmüş bir ders akış modeli ve modelin sonuçları ile ilgili detaylar verilecektir. Bu çalışmadaki amacımız, Türkiye şartlarına uygun ve donanımlı yazılım mühendisleri yetiştirmek için tasarladığımız yazılım mühendisliği dersi akış modelini ve buna bağlı olarak elde edilen analizin sonuçlarını diğer yazılım mühendisliği dersi eğitmenleri ile paylaşarak edindiğimiz tecrübelerimizden ve kazanımlarımızdan faydalanılabilmesini sağlamaktır.

Anahtar Kelimeler: Yazılım mühendisliği eğitimi, Vaka çalışması

A Course Flow Model Developed for Software Engineering Course

Abstract. As in the rest of the world, many software projects in our country exceed the planned budget and time limits, and the user expectations are not fully met. One of the important reasons for this is the lack of work force that well-trained in software engineering domain. In this article, details of a course flow model and the results of the model for the CENG 396 Software Engineering course, which has been carried out in Çankaya University Department of Computer Engineering for the last four years, will be given. The aim of this study is to explain the lesson flow model which is designed to cultivate the appropriate software engineering and software engineers equipped to conditions in Turkey. In addition, the secondary aim is to share the results of the analysis of this study with other software engineer course instructors so that they can benefit from our experience and achievements.

Keywords: Software engineering education, Case study

1 Giriş

Günümüzde, yazılım sektöründe çalışacak olan mühendislerden 10 veya 15 gün gibi kısa aralıklarda fonksiyonel yazılım ara ürünleri üretmeleri beklenmektedir. Bu bağlamda, genç mühendisin yazılım mühendisliği eğitimi sırasında ürünün farklı sürümleri üzerinde hatasız yazılım ürünü geliştirme yapması gerekmektedir. Ancak, yazılım mühendisliği eğitiminin temel hedeflerinde de belirtiği gibi amaç sürekli değişen araçlar konusunda bilgi vermek yerine yazılım mühendisliği işinin nasıl yapılacağı olmalıdır [6]. Özellikle, ACM dokümanında da belirtildiği gibi yazılım mühendisliği eğitimi öncelikle yazılım geliştirme sürecinin iyi bir şekilde öğretilmesine odaklanmalıdır [3]. Yazılım endüstrisinin çevik yazılım akımı sonrası ortaya çıkan ihtiyaçları göz önüne alındığında, verilecek eğitimin temel odağı, yazılım işlevlerinin ölçülmesi, öngörülmesi ve keşfedilmesi için kullanılan teknikler, metrikler ve araçlar ile yazılım tasarımının temel ilkelerini de kapsamalıdır.

Son zamanlarla, yazılım mühendisliği eğitimi sırasında, ürün geliştiren takımların teknik kabiliyetleri ile birlikte verimli iletişim kurabilme, takım ile uyumlu iş geliştirebilme ve takım ile üretebilme gibi sosyal kabiliyetlerinin de iyileştirilmesi ön plana çıkmaktadır [8]. Günümüzde, yazılım bakım ve onarım aktivitelerinin üretimden ayrılmaması gerekliliği doğrultusunda sistem üzerinde yapılan bir değişikliğin ardından çalışan sistemde oluşabilecek sıkıntıları belirlemek için sürekli entegrasyon metodolojisi ile üretim yapılmaktadır [4].

Yazılım Mühendisliği dersi kapsamında dikkate alınması gereken diğer bir konu ise, yazılımın üretimi sırasında elde edilen deneyimlerin öğrencilere aktarılmasını sağlamak, üretim faaliyetleri ve başarı kriterlerini belirlemek olmalıdır. Yazılım ürününün insan, kültür ve teknolojinin etkileşimiyle oluşan karmaşık ve sosyo-teknik yapısı dışında, gereksinimlerin değişkenliği ve geliştiren bireylerin bilgi ve tecrübe düzey farklarının oluşturabileceği sorunlar da öğrencilerle paylaşılması gereken temel bilgiler arasında olmalıdır. Yazılım mühendisliği eğitimi tartışmaları, yazılım mühendisliğinin doğuşundan bu yana edinilen tecrübeler doğrultusunda gelişerek iyileştirilmektedir. Ancak, yazılım mühendisliği eğitimi endüstriyel tecrübeye sahip, süreçler üzerinde bilgili ve deneyimli kişiler tarafından yapılmalıdır. Dolayısıyla, iyi bir eğitim sağlanması açısından diğer mühendisliklere göre daha pratiğe dönük bilgiler içeren bir eğitim yapısı benimsenmelidir.

Bu makalenin amacı araştırmacıların yazılım mühendisliği dersinde edindiği deneyim ve kazanımların paylaşımını sağlayarak bu dersi veren bilgisayar ve yazılım mühendisliği bölümlerine bilgi akışı sağlamaktır. Dersin çıktıları doğrultusunda öğrencilerin edinimleri ile ilgili olarak alınan geri beslemeler yarı yapılandırılmış görüşme tekniği kullanılarak raporlandırılacaktır. Bu makalenin devamı şu şekilde kurgulanmıştır: İkinci bölümde ülkemizdeki yazılım mühendisliği çalışmalarından bahsedilerek önerilen ders akış modeli açıklanacaktır. Üçüncü bölümde çalışmada kullanılan araştırma yönteminin detayları ve yapılan kalitatif vaka analizi açıklanarak sonuçları paylaşılacaktır. Dördüncü bölümde ise yapılan çalışmadan çıkarılan sonuçlar ile makale son bulacaktır.

2 Ülkemizdeki Durum ve Önerilen Akış Modeli

Ülkemizde birçok yazılım firması hala üretim süreçlerinden habersiz bir şekilde ürün geliştirmektedir. Bu şekilde yazılım geliştiren organizasyonlar, literatüre kazara geliştirme olarak geçmiş yaklaşımı benimsemiştir. Bu yaklaşım, müşteriden gelen gereksinimlere göre anlık olarak bir süreç veya ölçme yöntemi kullanmadan kod yazma şeklindedir. Böyle bir yöntem ile yazılımın geliştirilmesi ürünün testleri, bakımı ve kurulumu aşamalarında büyük problemlere sebep olmakla birlikte yazılım ürünleri sistematik bir süreç olmadan geliştirildiği için uzun vadede müşteri memnuniyeti sağlamak da mümkün olamamaktadır.

Yazılım üretim faaliyetleri, 1960'lardan bu yana farklı şekillerde ele alınmış ve yazılım geliştirme süreçlerini iyileştirmek için değişik yaklaşımlar önerilmiştir. Yazılım geliştirme metodolojileri gereksinim analizi, modelleme, ürünü kodlama, test etme ve ürünün bakımını içeren üretim basamakları açısından farklılık göstermese de bu evrelerin uygulama sırası, zamanlaması ve akış kontrolü açısından değişiklikler içerir. Klasik yazılım geliştirme (şelale yöntemi veya plan tabanlı geliştirme olarak da bilinir) yöntemi, bu evrelerin biri bitmeden diğer basamağın başlamasının ardışık bir şekilde icra edilmesi fikri üzerine kuruludur. Daha sonraları popüler olan yinelemeli yöntem, bu fazları kendi içinde küçük evrelere ayırarak geri besleme yardımıyla üretim verimini iyileştirmeyi hedefler. 2000'lerden sonra popüler olan çevik yöntemler ise yinelemeli yöntemleri arttırılmış evreler ve her yineleme sonunda çalışan yazılım ürün parçacığının üretim aktivitesinin

zaman kutucuklarına bölünmesi ilkesine dayanır [7]. Bu sayede, olası risklerle baş edilmesi ve çıkabilecek sorunların erken fark edilmesi hedeflenir. Yazılım geliştirme yöntemlerinden proje açısından en uygununa karar vermek, yazılım projesinin gereksinimlerine, geliştirecek grubun iş yapma kültürüne ve müşterinin beklentilerine göre seçilmelidir.

2000'li yılların başlarında, geleneksel yazılım geliştirme yöntemlerinin varolan projelerin sürekli değişen ihtiyaçlarını verimli bir şekilde adresleyememesinden dolayı, çevik yöntemler denilen bir grup metodoloji ortaya çıkmıştır. Bu yöntemler grubuna göre, yazılım projelerinin ilk aşamalarında detaylı bir sistem tasarımının yapılmasına gerek yoktur. Bunun yerine, çabuk şekilde, çalışan bir ara-sürüm elde etmek hedefiyle çalışılır. Böylece, erken risk tespiti ve potansiyel sorunların başlangıçta farkına varılması mümkün olmaktadır. Çevik yöntemlerin temel felsefesi, "*Gereksinim Belirsizlik*" İlkesi tarafından da desteklenmektedir: "*Yazılım sistemleri, kullanıcıları tarafından test edilmeden, yazılım gereksinimleri başarı ile tanımlanamaz*" [5].

Bir teslimatın kapsamının gerekli zamana göre tespit edildiği geleneksel metodolojilerden farklı olarak çevik yöntemler, zaman kutulu yinelemeler kullanır (diğer bir deyişle, aynı uzunlukta olan üretim aralıkları). Özellikle, proje riskleri, kapsamın bir serbest bırakma uzunluğu yaratmak yerine sınırlı bir zaman çerçevesinde tanımlanarak hafifletildiği anlamına gelir. Çevik yinelemeler, ürünün veya bir parçanın ürün geliştirme hattından teslim edilmesi için gerekli olan tüm aşamaları kapsamaktadır.

Bu bilgilerin ışığında, yazılım mühendisliği eğitimi, değişen koşullar ve yenilikçi metotları da göz önüne alarak teori ve pratik bilgilerin iyi bir şekilde harmanlandığı bir modele dayanmalıdır. Bu dayanak yazılım endüstrisindeki uzmanlar ve pratisyenler ile fikir ve bilgi paylaşımı içermeli, ders kapsamında yapılacak projelerin gerçek hayatta karşılaşılabilecek projelere benzer, yazılım mühendisliğinin üretim basamaklarına paralel olarak kurgulanmalıdır. Bu kurgu dersin doğasına uygun şekilde teoriden pratiğe geçişi kolaylaştıracak şekilde olmalıdır.

2.1 Yazılım Mühendisliği Dersi Akış Modeli

Ders, yazılım mühendisliği kavramları, yazılım geliştirme metodolojileri, yazılım aşamaları, gereksinim analizi, yazılım testleri ve proje yönetimi ile ilgili olarak temel kavramları anlatmaktadır. Ders kapsamında hem bireysel hem de takım tabanlı aktiviteler bulunmaktadır.

Dersin ana çıktıları şu şekilde tanımlanmıştır. Öğrenci ders kapsamındaki aktivitelerle,

- Yazılım mühendisliğinin temel kavramlarını anlar.
- Yazılım gereksinimlerini değerlendirir, analiz eder ve tasarlar.
- Yazılım tasarım tekniklerini uygulama becerisi kazanır.
- Şelale, Yinelemeli (iterative) geliştirme, Scrum, Kanban ve XP gibi plana dayalı ve çevik metodolojilerinin artılarını ve eksilerini kıyaslar ve karşılaştırır.

– İlgi alanına bağlı olarak, yazılım mühendisliğindeki yeni trendler, araştırma konuları ve endüstride kullanılan teknolojiler hakkında bilgi edinir.

Tablo 1'de dersin 14 haftaya yayılmış konu başlıkları görülmektedir. Öncelikli olarak öğrenciye mühendisliğin ne demek olduğu ve bilgisayar bilimleri, bilgisayar mühendisliği ve yazılım mühendisliği konuları arasındaki farklar anlatılmaktadır. Daha sonra, yazılım geliştirirken karşılaşılacak hata türleri ve bu hataların oluşmaması için alınması gereken önlemler ile birlikte özellikle ihtiyaç analizi ve testlerin önemi vurgulanmaktadır. Bu konuları takiben, yazılım süreç modelleri klasik yöntemlerden çevik yöntemlere doğru şekilde açıklanmaktadır. Bu kısımda, özellikle eğitmen kendi iş tecrübesini öğrencilerle paylaşarak pratik ile teorik kurgular arasında bir köprü kurmaktadır.

Tablo 1. CENG 396 Yazılım Mühendisliği Dersi İçeriği [1]

Hafta	**İçerik**
1	Modül Tanıtımı ve Giriş
2	Yazılım Mühendisliği'ne Giriş
3	Yazılım Hataları, Arızalar ve Etiği
4	Yazılım Süreç Modelleri
5	Ampirik Manzara ve Kullanım Senaryoları
6	Gereksinim Analizi
7	Geleneksel ve Çevik Yazılım Geliştirme (XP,SCRUM)
8	Geleneksel ve Çevik Yazılım Geliştirme (KANBAN)
9	Yazılım Tasarımı ve Mimarisi I
10	Yazılım Tasarımı ve Mimarisi II
11	Yazılım Testleri
12	Yazılım Süreç Yönetimi
13	Dönem Sonu Sunumları
14	Dönem Sonu Sunumları

Öğrenci, UML kullanarak ihtiyaç analizi yapmayı öğrendikten sonra yazılım mimarisi ve yazılım testleri konusunda temel bilgiler edinir. Edindiği bu bilgileri üç aşamalı olarak sunulan bir takım projesi ile gerçek yazılım projelerinde yapıldığı gibi ekip çalışması yaparak pratik halde dönüştürür. Bu ders kapsamında düşünülüp planlanmış ödevleri ise şu şekilde sınıflandırabiliriz.

1. Europass [2] formatında özgeçmiş hazırlama ve seçilen bir gerçek işe başvuru ödevi.
2. Endüstrideki bir şirket ziyaretine dayanan yazılım süreç değerlendirme ödevi.
3. Yazılım ihtiyaç analizi, tasarım ve test kavramlarını kapsayan üç aşamalı bir grup proje ödevi.

4. Yeni teknolojilerin tanınması için yapılan bir dönem ödevi.

Ders, ödev ve proje ağırlıklı bir ders olarak, öğrencinin özel iş hayatında en çok gerek duyacağı bağımsız ve grup halinde çalışabilme kabiliyetini geliştirmeyi hedeflemektedir. Bu doğrultuda dersin not verme ölçeği şu şekildedir. (1) Ödevler %40; (2) Ara sınav %15; (3) Dönem Ödevi %15 ve (4) Final sınavı %30. Dönem ödevi ayrıca kendi içinde sunum ve doküman tasyon kısmından oluşmaktadır. Sunum %40 doküman tasyon notu da %60 üzerinden verilmektedir.

Öğrenci dersin ilk haftalarında özellikle bilgisayar ve yazılım mühendisliği arasındaki farkları irdelerken bir yandan da iş başvurusu yapmak için neler yapması gerektiği konusunda bilgilendirilmektedir. Bu bağlamda öğrencilerden, Avrupa standartlarına uygun, Europass [2] formatında bir özgeçmiş ve seçtiği bir iş için ön yazı hazırlaması beklenmektedir. Bu aktivite sayesinde öğrenci hangi tür işlere merak duyduğunu keşfetme fırsatı bulur. Bu fırsat özellikle öğrenciye dönem ödevi için konu verirken göz önüne alınmaktadır. İkinci ödev olarak, öğrenci özellikle yazılım süreçlerinin pratik boyutunu tanımak ve yazılım şirketleri ile ilişki kurmak için yazılım geliştirilen organizasyonlara yönlendirilir. Bu sürede kendisinden gittiği organizasyonda yazılım süreçlerinin nasıl uygulandığı veya yazılım mühendisliği etiği gibi önemli konularda yazılım uzmanları ile görüşerek bilgi toplaması ve bunu raporlaması istenir. Daha sonra grup ödevi olarak nitelendirilen, üç aşamalı; ihtiyaç analizi, tasarım ve test ödevi başlar. Bu ödev öncelikle, 2 veya 3 kişilik gruplara ayrılmış öğrencilerin ihtiyaç dokümanını analiz ederek işlevsel ve işlevsel olmayan ihtiyaçları bulmasıyla başlar. Bu ihtiyaçlar doğrultusunda ikinci aşamada ise öğrenci grubundan proje ile ilgili tasarım yapması beklenir. Son aşamada da öğrenci grubundan elde edilen ihtiyaçlar doğrultusunda 10 tane test senaryosu ("*test case*") yazması beklenir. Son ödev olarak öğrencinin seçtiği bir konuda alan araştırması yapması istenir. Bu şekilde öğrencilerin ilgilendiği konular üzerinde yeni teknolojileri tanıma ve inceleme fırsatı bulması sağlanır.

3 Yöntem

Nitel araştırma insanı enstrüman olarak kabul eder. Bu teknik, katılımcıların davranışlarını yorumlayıcı anlam kalıpları içinde araştırmayı amaçlayan natürel bir yaklaşımdır. Katılımcıların kendi sözel anlatımlarından çıkartılan zihinsel tutum ve sosyal davranışları inceleyerek doğal çalışma ortamlarında elde edilen bilgiler halinde sistematik olarak sınıflandırır. Nitel araştırmacılar, insanların değer sistemleri ve düşünceleri hakkında veri toplamak için mülakat, bireysel deneyim çıkarımı, vaka analizi ve odak grupları gibi teknikleri kullanmaktadırlar. Toplanan veriler, sözcükleri yakından incelemek ve katılımcıların görüşlerinin daha ileri denetimini yapmak için bağlamsal bir çerçeve içinde raporlandırılır. Genel olarak, nitel yöntemler tümevarımsal yöntemler olarak kabul edilir. Bir teori üretmek için yeni veya keşfedilmemiş bir fenomeni araştırmak için sıkça kullanılırlar. Özellikle, araştırmacıların, grup etkileşimleri gerektiren derinlemesine cevaplar veya araştırmalar (örneğin, yanıt veren ve araştırmacı olan kişiler arasındaki etkileşimler) aramak için katılımcılarla etkileşime girmesi gerektiği durumlarda bu

araştırma yönteminden faydalanırlar. Bununla birlikte, tipik olarak nitel araştırmalar küçük gruplar halinde veya sınırlı sayıda katılımcı ile yürütülmektedir. Çalışma sonucunda elde edilen bulgular üzerinden analizler yapılır, çıkarımlar katılımcılarla veya diğer araştırmacılarla paylaşılır.

Yazılım mühendisliğinde yapılan bilimsel araştırmaların bir amacı, yazılım yaşam döngüsünü incelemek, şirketlerde olup biten olayları sistematik yöntemler ışığında anlayabilmek ve daha önemlisi gerektiği zaman müdahale edebilmektir. Yazılım süreçleri çalıştırılırken diğer firmalara aktarılabilecek bilgiler çok farklı şekillerde gözlenebilmektedir. Bu bilgilerin araştırma desenlerinin bize tanıdığı imkanlar doğrultusunda sınıflandırılması gerekmektedir. Dolayısıyla yazılım geliştirme aktiviteleri sırasında ortaya çıkan, doğası ve özellikleri bakımından birbirlerine benzemediği düşünülen karmaşık olaylar farklılıkları ve benzerlikleri göz önüne alınarak sınıflandırılmalı, bu olaylar karşısında sağlanan başarılar yazılı hale getirilerek, elde edilen bilgi diğer araştırmacılara ve endüstriye sunulmalıdır. Örnek olay inceleme yöntemi, belli bir konu ile ilgili olarak karşılaştığımız sorunların sınıflandırılması ilkesine dayanır. Yazılım yaşam döngüsünün canlı bir varlık gibi düşünülmesi, gerek kişisel problemlerin anlaşılması gerekse elde edilen bu sorunların çözümlesi açısından önemlidir. Bu sayede yazılım geliştirme süreçlerinde gözlenen durumlar mercek altına alınarak incelenen vakalardan çıkartılan derslerin aktarılabilmesi sağlanır. Doğal bağlamlarında açık uçlu bir şekilde izlenen, yalnızca tek bir bakış açısından kaçınılması, örnek olayların özelliklerini, yazılım geliştirme kalıpları, takım yapılarının ya da uygulanan yazılım geliştirme süreçlerinin analitik yöntemlerle karşılaştırılmasından yararlanarak uygulanması önerilir. Ayrıca karmaşıklıkların tanımını ve açıklamasını yapabilmek karşılaştırılabilmeleri açısından önemlidir. Yin [9], vaka incelemelerine uygun altı farklı veri kaynağı olduğunu iddia etmektedir; (i) belgeler, (ii) arşiv kayıtları, (iii) röportajlar (veya anketler), (iv) doğrudan gözlem, (v) katılımcı gözlemi, (vi) fiziksel eserler. Bu kaynaklar, tek bir kaynak olarak veya birbirlerini tamamlayıcı olarak kullanılabilirler. Tablo 2'de, verilerin kaynakları, güçlülükleri ve zayıf yönleri gösterilmektedir.

3.1 Kalitatif Vaka Çalışması

Bu çalışmada yarı yapılandırılmış görüşme tekniği kullanılarak 2013 - 2017 yıllarında CENG 396 dersini almış ve iş hayatına atılmış on beş öğrenci ile temas kurulmuş ve bu öğrencilerden on ikisi ile röportaj yapılmıştır. Bu çalışmaya şu ön bilgilendirme e-postası ile başlanmıştır. "*Yazılım Mühendisliği dersinin önemini üçüncü sınıf öğrencilerine göstermek için dersi benden alan arkadaşların görüşleri doğrultusunda bir tanıtım videosu hazırlamak istiyoruz. Bunun için akıllı telefonunuzla sadece 3 soruya 3 dakikayı geçmeyecek şekilde bir video çekmenizi rica ediyorum.*

1. Adınız, Soyadınız, Hangi yıl mezun olduğunuz

2. Yazılım Mühendisliği dersinde öğrendikleriniz iş hayatında karşınıza nasıl çıktı ve ne işe yaradı.

3. Dersi alan arkadaşlara neler tavsiye edersiniz."

Tablo 2. Olay Araştırmasında Veri Toplamak için Kullanılan Kaynaklar

Veri Kaynakları	Güçlü Yönler	Zayıf Yönler
Dokümantasyon	• Stabil – Tekrarlanabilir gözden geçirme • Göze batmayan - Vaka çalışmasından önce bulunan • Net, isim ve olayların açıklamaları	• Edinilebilmesi güç • Yanlı seçim olasılığı • Araştırmacı önyargısı olasılığı • Sürekli erişim engeli
Arşiv Kayıtları	• Yukarıdaki güçlü yönler • Kantitatif	• Yukarıdaki zayıf yönler • Gizlilik ve erişim engeli
Röportaj ve Anket	• Hedefli – Vaka çalışması konusunu adresleyebilir • Anlaşılabilir – Neden sonuç ilişkilerini anlatabilir	• Yanlı cevaplar içerebilir • Anketör istediklerini duyuyor olabilir
Direkt Gözlem	• Gerçeklik – Vakaları gerçek zamanlı olarak inceleyebilir • Bağlamsal – Olayları o bağlam içinde izleyebilir	• Zaman alıcı • Seçicilik bazı gerçekleri perdeleyebilir • Refleksivite - Gözlemcinin varlığı sonuçları etkileyebilir
Kişi Gözlemi	• Yukarıdaki güçlü yönler • Davranış ve kişisel arası ilişkilerin gözlenmesi	• Yukarıdaki zayıf yönler • Araştırmacının hareketleri sonucu etkileyebilir
Fiziksel Eserler	• Kültürel özellikler ile ilgili anlayış • Teknik operasyonlar ile ilgili anlayış	• Seçicilik • Uygunluk

Katılımcılardan toplanan bilgiler 20 sayfalık bir dokümana dönüştürülmüştür. Katılımcıların yorumlarından çeşitli alıntılar şu şekildedir.

Katılımcılardan birisi:

> *" Yazılım Mühendisliği dersinin öneminden biraz bahsetmem gerekirse aslında ilk aşamada hem sözlü hem de yazılı mülakatlarda burada öğrendiğim süreçler ve bilgiler mutlaka sizin karşınıza çıkıyor. Ve şu an ki işyerime giriş aşamasında yapılan sözlü ve yazılı mülakatlarda burada öğrendiğim bilgiler karşıma çıktı. Onun dışında sadece mülakatlar için değil profesyonel hayatta da sizden yazılım mühendisliğinde anlatılan süreçleri bilmeniz ve uygulamanız bekleniyor. O yüzden daha iyi bir mühendis olabilmeniz adına gerçekten çok önemli konular. "*

Katılımcılardan bir diğeri:

" İşe başladığım zaman ilk olarak üzerinde çalışacak olduğum projenin yazılım gereksinim analizlerini yaptım. Daha sonra derste görecek olacağınız software requirement spesification SRS dokümanı, SDD dokümanı hazırlamıştım, daha sonra gereksinimlere ait use-case'leri hazırladım. Ve bunların hepsini dokümante etmiştim... ...yazılım geliştirme metodolojilerinden SCRUM'ı katı kuralları ile birlikte uyguluyoruz aslında. Derste de göreceksiniz Scrum'ın özelliklerini iki haftada bir sprint planning toplantıları yapıyoruz iki haftalık süreçlerle sprint'lerimiz oluyor. Bu planinglerde bu bir sonraki sprint'e gelene kadar hangi gereksinimleri tamamlamamız gerektiğini nasıl tamamlarız grup içinde takım içinde konuşup puanlayıp sprinti başlatıyoruz. İki haftalık süreçte bunları tamamlamaya çalışıyoruz... ...ve görmüş olduğunuz gibi software engineering dersinde görecek olacağınız her şey iş hayatınızda kesinlikle ve kesinlikle karşınıza çıkıyor. Bunun çok büyük yararını da emin olun görüyorsunuz. Software engineering dersini donanımlı bir şekilde bitirmek sizler için güzel bir avantaj olacak. Benim size tavsiyem bu ders boyunc haftalık ödevleri ve dönem projelerini kesinlikle dikkatli bir şekilde tamamlıyor olmanız. Ben bunların kendim için çok büyük yararını gördüm sizlere için de yararlı olacağını düşünüyorum. "

Diğer bir röportajda ise:

" Yazılım mühendisliği dersi sayesinde X firmasındaki işim ilk tecrübem olmasına rağmen UML diagram flow chart nedir nasıl çizilir requirement analiz nasıl yapılır software development metodolojileri nelerdir gibi pek çok konuyu aslında bilerek işe başlamıştım bu benim için büyük bir avantajdı. Çünkü hala pek çok üniversitede bu konuları ele alan dersler ne yazık ki müfredatta bulunmuyor ve ben bu bilgiler sayesinde işe adaptasyon sürecini bir an önce tamamlayıp işime odaklanabilmiştim. Şu an çalıştığım projede birkaç satırlık kod değişimlerinde dahi onlarca sayfalık doküman hazırlayıp bunları stakeholder'lara sistem artitechlere review için gönderiyoruz. Ve bu dokümantasyon sürecinde derste öğrendiğim hemen hemen her şeyi kullandığımı belirtmeden geçemeyeceğim. Dört yıllık eğitim hayatıma baktığımda software engineering dersinin beni profesyonel hayata hazırlayan en önemli derslerden biri olduğunu açık yüreklilikle dile getirebilirim. Bu noktada sizlerde derste yapacağınız projeleri ciddiye alıp bilgi birikimizi arttırmayı hedeflerseniz iş hayatında mutlaka birkaç adım öne geçeceksiniz. "

Diğer bir yorum:

" Mülakatların hepsinin kendi teknik domainleri var evet ama software arcitech (software engineer dersi olarak) her mülakatımda karşıma gelen sorular vardı. Mesela bir mülakatımda direkt olarak choesion ve coupling soruldu. Diğer bir mülakatımda agile development process hakkındaki düşüncelerim soruldu. Bütün mülakatlarımda bu oldu şöyle; bir ürün veya projeyi hangi süreçlerden geçirerek son ürünü elde ettiğim soruluyor...

...Standart ve dokümantasyonu olmayan işler sadece çalışma olarak adlandırılıyor iş hayatında. Eğer standart ve dokümantasyonu varsa ve büyün stepleri de uygulanmışsa artık bu bir ürün haline dönüşmüş oluyor. Biz şu an iş yerimizde SCRUM uyguluyoruz. Bu Scrum'ı da derste görmüşsünüzdür. Bir süreç tanıyorsunuz artık sürecin ne olduğunu biliyorsunuz ve süreç ve software engineering dersini bilen mühendisler aralarında sadece bu jargon üzerinde konuşuyor... ...kimse sizin yazdığınız kod satırı ile gerçekten ilgilenmiyor. Yaptığınız işin profesyonel anlamda değer taşıması için bu kalıplara standartlara uyması gerekiyor. "

Diğer bir katılımcı ise:

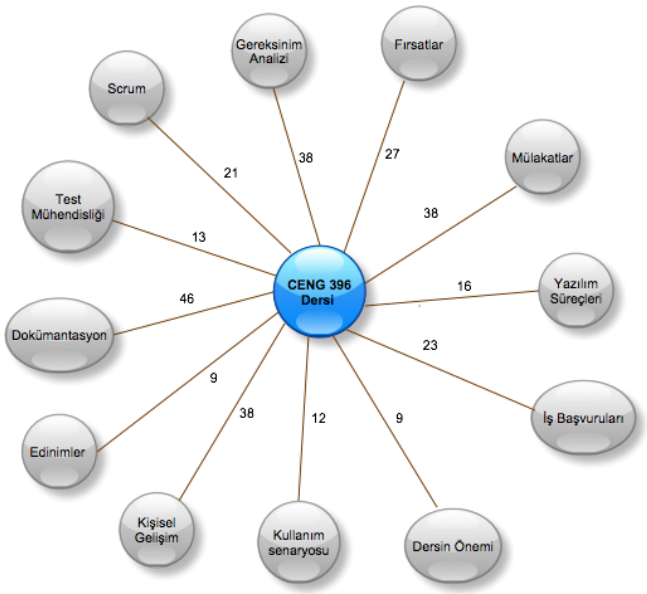
" Yazılım mühendisliği dersinde öğrenmiş olduğumuz konular tamamen iş yaşamına yönelik konular çünkü iş yaşamında da bir projenin başlangıç ve bitiş süreleri var, projenin aşamaları var bu aşamalarda olan bir çok insan var. Mesela bu derste öğrenmiş olduğumuz analiz nedir, analizin nasıl yapılacağı, analiz dokümanını nasıl hazırlanacağı ya da design ile alakalı design'ın nasıl yapılması gerektiği design'ın dokümantasyonu ya da test süreçlerinde neden test yapıyoruz nasıl test yapıyoruz hangi tür testleri kullanıyoruz testing dokümantasyonunu nasıl yapıyoruz senaryoları nasıl hazırlıyoruz bunlar tamamen iş yaşamında bizi bir tık daha ileri götürebilecek konular. Çünkü, iş yaşamına girdiğimizde bu konuları kavramak birazcık zaman aldığı için biz önceden öğrenerek kulak dolgunluğuyla gittiğimiz için üstüne daha çok koyup daha çok kendimizi gösterebiliyoruz. Mülakatlarda da süreci bilmek gerçekten çok önemli süreci bilen mühendislerin bir adım önde olduğunu düşünüyorum. Hem süreci bilmek hem bu sürecin içende nerede yer alacağını bilmek iş yaşamında çok önemli. Bundan dolayı bu yazılım mühendisliği dersinde öğrenebileceğiniz ne varsa örneklerle konuları tamamen anlayarak öğrenmeniz gerçekten iş yaşamına yatırım olur sizin için. "

Başka bir katkı sağlayan öğrenci:

" Örneğin mezun oldunuz. Bir iş görüşmesine gittiniz. Görüşme sonunda karşı taraf sizin bir sorunuz var mı diye size bir soru yönelttiğinde projelerinizi geliştirirken kullandığınız metodolojiler neler diye sorarak oradan bir güzel yürüyüp karşı tarafa bir şeyler bildiğinizi gösterme fırsatı yakalayabilirsiniz. Bu şekilde de karşı tarafta güzel bir izlenim bırakmış olursunuz diye düşünüyorum. Size naçizane tavsiyem agile yöntemlerini iyi bir şekilde kavrayın. Bildiklerinizi iş mülakatı sırasında karşı tarafa mutlaka satın... ...bir şirkete girdiğinizde bir projeye dahil edildiğinizde ne yapmanız gerektiğini bilmekten önce aslında nasıl yapmanız gerektiğini bilmeniz oldukça büyük önem arz ediyor çünkü başlarda öğrenciyken ben de sadece kod'un bir şeyler için yeterli olacağını düşünmüştüm ancak kod yazmak projenin oldukça ufak bir parçasını oluşturduğunu deneyimledim. "

Yapılan tüm yorumlar tematik içerik analizi yöntemiyle incelenmiştir. Burada amaç, öğrencilerle yapılan röportajlarda toplanan bulgulara dayalı olarak

birbirine benzeyen verileri belirli kavramlar ve temalar çerçevesinde bir araya getirmek ve düzenlemektir. Çalışma sırasında elde edilen dokümanlar sistematik olarak incelenmiştir. Ayrıca, kullanılan ifade ve kavramlar da uygun temalara yerleştirilerek analiz edilmiştir. Toplanan bilgiler Nvivo 8 paket programı yardımıyla, içerik analizi yöntemi kullanılarak tema ve kategoriler altında değerlendirilmiştir. Araştırmacılar, tematik analiz çerçevesi ile ana temanın diğer alt temalar altında incelenebileceğine karar vermişlerdir. Araştırma sonucunda ana tema ve 12 alt temaya ulaşılmıştır. Çankaya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde zorunlu ders olarak verilen CENG 396 yazılım mühendisliği dersi için alınan yorumlarda 290 değer ifadesi bulunmuştur. Bu ifadeler ışığında ana ve alt temalar ve bu ifadelerin frekans değerleri Şekil 1'de sunulmuştur.



Şekil 1. Öğrencilerin Ders ile ilgili Değer İfadelerine İlişkin Oluşturulan Ana Temalar ve Frekans Değerleri

Dersi alan öğrencilerin röportaj metinlerinde geçen değer ifadeleri incelendiğinde "dokümantasyon" (46) en çok temasına yer verildiği görülmüştür. "Kişisel gelişim ve katkıları", "gereksinim analizinin önemi" ve "dersin mülakatlarda olan faydaları" temalarına toplamda 38 kez rastlanmıştır. Bu ders ile ilgili konuşmalarda değer ifadesi olarak "iş başvuruları" (23) ve dersin iş başvurularına olan pozitif etkisini temasını işleyen "fırsatlar" (27), "Scrum" (21), "yazılım süreçleri" (16) temalarına sıkça yer verilmiştir. Ayrıca, bu ifadeyi, "dersin önemi" (9), "edinimler" (9) kavramları takip etmektedir. Ayrıca, sürekli entegrasyon kavramının ortaya çıkması ile anlamsal olarak önem kazanan "test mühendis-

liği" ifadesine ise 13 kez rastlanmıştır. Bu bulgular öğrencilerin şirketlerle karşılaştıktan sonra kendilerine derste verilen bir çok kavram ile karşılaştıklarını ve bunların önemini fark ettiklerini göstermektedir. Özellikle dersin kişisel gelişimlerine katkıları, dersteki edinimlerinden bahsetmeleri, mülakatlarda dersin içeriğinden faydalandıkları fikrine sıkça rastlanması dersin olabildiğince faydalı olduğu görüşünü doğrulamaktadır.

4 Sonuçlar

Bu makalede Çankaya Üniversitesi Bilgisayar Mühendisliği programında zorunlu olarak verilen CENG 396 Yazılım Mühendisliği dersinde verilen eğitim ile ilgili yürütülen faaliyetlerin bir araştırma raporu sunulmuştur. Amacımız, edindiğimiz deneyimleri öğrencilerden gelen geri beslemeler doğrultusunda diğer üniversiteler ile paylaşmak ve bu konularda edinimlerimizden faydalanmalarını sağlamaktır. Dersin mümkünse yazılım endüstrisinde çalışmış veya yazılım mühendisliği konusunda danışmanlık yapacak bilgi birikime sahip tecrübeli eğitmenler tarafından verilmesi ve sürekli yeni çıkan süreçler ve ilgili teknolojiler ışığında geliştirilmesi gereklidir. Öğrencilerin bu ders kapsamında hem yazılım mühendisliğinin temellerini öğrenmelerinin hem de yazılım endüstrisine hazırlanmalarının olumlu sonuçları makalede paylaşılmıştır. Öğrenciler temel yazılım mühendisliği bilgisinin yanında mülakatlarda da sıkça karşılaştıkları kavramları derste gördüklerinden dolayı iş başvurularında avantaj sağlamıştır. Yazılım mühendisliği dersi çerçevesinde öğrencilerin ve iş piyasasının isteklerini dikkate almak ve onlarla işbirliği içinde olmanın olumlu sonuçlar doğurduğu açıkça gözlenmektedir.

Kaynaklar

1. CENG 396. http://ceng396.cankaya.edu.tr/ (2018), [Online; accessed 12-June-2018]
2. Europass. https://europass.cedefop.europa.eu/ (2018), [Online; accessed 12-June-2018]
3. Ardis, M., Budgen, D., Hislop, G., Offutt, J., Sebern, M., Visser, W.: Software engineering 2014: Curriculum guidelines for undergraduate degree programs in software engineering. joint effort of the ACM and the IEEE-Computer Society (2014)
4. Fowler, M., Foemmel, M.: Continuous integration. Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf 122, 14 (2006)
5. Humphrey, W.S.: A discipline for software engineering. Addison-Wesley Longman Publishing Co., Inc. (1995)
6. Inverardi, P., Jazayeri, M.: Software Engineering Education in the Modern Age: Software Education and Training Sessions at the International Conference, on Software Engineering, ICSE 2005, St. Louis, MO, USA, May 15-21, 2005, Revised Lectures, vol. 4309. Springer (2006)
7. Schwaber, K., Beedle, M.: Agile software development with Scrum, vol. 1. Prentice Hall Upper Saddle River (2002)
8. Yılmaz, M., Taşel, S., Güleç, U., Sopaoğlu, U.: Bilgisayar mühendisliği bitirme projeleri için düşünülmüş bir süreç yönetim modeli. 10th Turkish National Software Engineering Symposium (UYMS 2016) (2016)
9. Yin, R.K.: Validity and generalization in future case study evaluations. Evaluation 19(3), 321–332 (2013)