Yazılım Mühendisliği Eğitim Öğretimi İçinde Yazılım Kulubeleri ve Yazılım Kalite Kontrol Elemanı Kullanımı

Ümit M. Karakaş, Uğur Ayan
İstanbul Kültür Üniversitesi, Bilgisayar Mühendisliği Bölümü, İstanbul
{u.karakas, u.ayan}@iku.edu.tr

Özetçe

Bilgisayar Mühendisliği bölüm programı içinde Yazılım Mühendisliği (Software Engineering Course) dersinde, Yazılım Kulubesi (Software Hut) yöntemi ile 28 yazılım kulubesinden elde ettiğimiz ve henüz elde edemediğimiz amaçlar özetlenmektedir. Yazılım Kulubesi içinde, not ortalaması en yüksek öğrenciyi Yazılım Kalite Kontrol Elemanı olarak atamak son yıllardaki başarılı buluşumuz olarak görünmektedir. Elde etmiş olduğumuz amaçlar arasında, "Yazılım Kalite Kontrol Elemanı" kavramının pekiştirilmesi, üç akademik yılda öğrenci başına 0.5 KDSI den 1.0 KDSI ye yükselen "yazılım üretkenliği", "çok dilli kullanıcı arabirimi (Multi-Lingual User Interface Design & Implementation)" bulunmaktadır. Öğrenci projelerinde, üç akademik yılda KDSI başına 5.26 hata oranından 2.64 hata oranına iniş ulaşabildiğimiz düzeydir.

Anahtar sözcükler *(Türkçe): Yazılım Kulubesi, Yazılım Kalitesi, Yazılım Üretkenliği, Yazılım Kalite Kontrol Elemanı, Yazılım Mühendisliği Dersi,*

Abstract

This paper summarize our achievement and still unachieved aims while teaching Software Engineering Course (SWE-course) within curriculum of Computer Engineering by using Software Hut approach (28 Software Hut) for student projects. Adding the top GPA students as a Software Quality Auditor to the Software Hut seems our successful modification to Software Hut approach. Our achievements included in adding Software Quality Auditor to Software Hut, increasing Software Productivity from 0.5 KDSI to 1.0 KDSI per student, promoting Multi-Lingual User Interface Design & Implementation. Error rate per KDSI is reduced from 5.26 errors per KDSI to 2.64 errors per KDSI.

Keywords *(English): Software Hut, Software Quality, Software Productivity, Software Quality Auditor, Software Engineering course (SWE-course),*

1. Giriş

Türkiye'de Bilişim ve Bilgisayar alanında ilk iki lisans programı 1977'de eş zamanlı olarak başlamıştır *(ODTÜ, Hacettepe)*. 28 yıl sonra, Ağustos 2005 de, bu alandaki Türkiye ve Kuzey Kıbrıs bölümlerin 54'ü (%93'ü) Bilgisayar Mühendisliği, 3'ü (%5'i) Yazılım Mühendisliği idi[1],[2]. Ağustos 2009 da, dağılım 78 program ile (%80.0) Bilgisayar Mühendisliği, 11 program ile (%11.5) Yazılım Mühendisliği, ve 8 program ile (%8.5) oranında diğerlerinden oluşmuştur.. Türkiye'de "Yazılım Mühendisliği (Software Engineering)" hız kazanmaktadır.

ABD de Bagert, 2003 yılında ABD ve Kanada'da 22 önemli üniversitenin "Yazılım Mühendisliği (Software Engineering)" lisans öğrencisi kaydettiğini [3] belirtmiştir.

İngiltere'de bu yıl 121 Üniversite bulunmaktadır. *(Bu sayıma İngiltere'de yerleşkesi (Kampus) bulunan 11 yabancı üniversite ve 13 "University College" dahil değildir)* Yazılım Mühendisliği (Software Engineering) (SWE-bs) bölümü sayısı 62, İngiliz Üniversiteleri arasında bölümün bulunma olasılığı %50 dir.

Birinci yazar, 2003 de "Yazılım Mühendisliği artık bir üniversite dersi adı değildir! Yazılım Mühendisliği artık bir Mühendislik Bölümü adıdır" başlıklı bildirisini[4] yayınlanmıştır. Bu bildirideki temel nokta, "Yazılım Mühendisliği (SWE-Bs)" nin artık "Elektrik Mühendisliği", "Elektronik

Mühendisliği", "Makine Mühendisliği" gibi "lisans programı adı" haline geldiğinin belirlenmesidir[4], "Yazılım Mühendisliği dersi (SWE-course)" adından vazgeçilmesi önerilmiştir.

Birinci yazar, Prof.Taylı ile 2006 da, "AB uyum sürecindeki TC, {bilgisayar lisans || yazılım mühendisi} karmasını Bilgisayar Mühendisliği etiketi ile pazarlamaktan vazgeçmeli, ACM & IEEE-CS curriculum committee'nin tanımlarına doğru dengeli bir açılımı hemen yapılmalıdır"ı[5] önermiştir.

Aşağıda paylaşılacak olan mesleki deneyim Kültür Üniversitesi Bilgisayar Mühendisliği programı içinde yer almaktadır. Bu bölümün programı, Ekim 2004 de düzeltilmiş IEEE-CS & ACM ccCE2004 e oldukça yaklaştırılmıştır.

Bilişim alanında bazı akademisyenlerin işaret etti-ği bir nokta "lisans mezunlarının önemli bir kesi-minin 1.0 KDSI *(Kilo Delivered Source Instruction)* dahi program üretmeden mezun oldukları, oysa güncel yazılım boylarının 100.0+ KDSI olduğudur. Bu durum bu alandaki öğretim çıktısı ile iş gereksinimi arasında farkın giderek açılması yönündedir"

En az üç yıl önce başlayan savımız, TC İstanbul Kültür Üniversitesi Bilgisayar Mühendisliği mezunlarının, en az 1.0 KDSI yazılımı yedinci yarıyılda üretmiş olmaları ve sekizinci yarıyıldaki 3 – 6 KDSI lik mezuniyet projelerine başlarken

a) "Yazılım Ölçümü (Software Meausurement)" ile ilk kavramları almış olmaları

b)"Yazılım Kalitesi (Software Quality)" ile ilk kavramları almış olmaları,

c) CSE042 Software Engineering dersi laboratuarı (1 kredi eşdeğeri) için 1.0 KDSI üretmiş olmaları'na dayanmaktadır.

Bunun, Türkiye liselerinin mezunlarının, %10.8 i ile orta noktası modellenebilecek IKU-BM öğrenci girdisi ile mümkün olduğu başlangıç savımızdır.

Bunu sağlayabilmek için, bölüm programındaki "Yazılım Mühendisliği" dersi (SWE-course) Ekim 2007 den itibaren sekizinci yarıyıldan yedinci yarıyıla aktarılmıştır.

Bu durumda, bu ders (CSE042 Software Engineering) (SWE-course) içinde beklediğimiz gelişme dört noktada yoğunlaşmaktadır:

Amaç-1: Daha önceki derslerde öğretilmesi ihmal edilmiş, ders içeriklerine konulmamış gerekli mesleki standardların (ISO, IEEE-Std) öğretilmesi,

Amaç-2: "Yazılım Mühendisliğine Giriş" temel kavramlarının verilmesi,

Amaç-3: bizim alanımızda onyıllardır kullanılmış ve kullanılmaya devam eden "Yazılım Kulubeleri (Software Hut)" [6] yaklaşımı ile öğrencilere takım halinde yazılım geliştirme alışkanlığı kazandırma.. Bu alanda kavramlar karıştığında, 1991 de CMU'da Mary Shaw & James Tomayko'nun[7] beş altkümeye ayırarak sınıflamış olduğu "yazılım mühendisliği ders projesi"ni bu beş düzeyin ortadaki üçüncü öğesi biçiminde vermek benimsediğimiz yoldur,

Amaç-4: "Yazılım Üretkenliği" (Software Productivity) ve "Yazılım Kalitesi" (Software Quality) ölçümlerini en az üç yıl sure ile yaptıktan sonra yayınlamaya başlamaktır.

2. İlk İki Amacımızın Kısa Tanımı

Bilgisayar Mühendisliği, Yazılım Mühendisliği alanında Uluslararası Standardlar Örgütü (ISO) ve Uluslararası Elektrik Elektronik Mühendisleri-Bilgisayar Grubu'nun (IEEE-CS) birçok standardı bulunmasına karşılık, ilgili standardların Bilgisa-yar Mühendisliği ders kitaplarında genellikle ihmal edildiği, bu standardları bilen öğrenci ve öğretim üyesi frekansının düşük olduğu gözlenmiştir. Öğrenciye "Sistem Çözümleme Vizyonları"nın ilk ikisi karikatürize edilerek öğretil-mektedir. (SCVn: Sistem Çözümleme Vizyonu). Öteki deyişle, karikatürize edilen ilk iki sistem çözümleme yaklaşımdan öğrencinin psikolojik olarak uzaklaşması sağlanmaktadır.

SÇV1 : Bu sistem çözümlemenin vizyonu benim masam ve benim aklım ile sınırlıdır. *Veri sözlüğü kodları (data dictionary codes) ve veri öğelerinin (data item) değer sınırları için hemen aklıma gelen tanımı yaparım.*

SÇV2: Bu sistem çözümleme vizyonu bizimle aynı alanda çalışan kuruma kadar uzanır. *"Türkiye'de bizim alanımızda çalışan kurumlar bu veri sözlüğü için ne yapmışlar, bir bakalım" bu Sistem Çözümleme Vizyonunun yaklaşımıdır.*

SÇV3: Biz uluslararası standarda uyarak veri söz-lüklerimizi hazırlayalım. *Bazı veri sözlükleri Türkiye'nin ihtiyacını karşılamıyor ama Türkiye*

içinden birileri ilgilenip düzelttiği zaman biz de onları izleyerek düzeltiriz.

SÇV4: "Uluslararası standarda uyarak veri sözlüklerimizi hazırlayalım. Bazı veri sözlükleri Türkiye'nin ihtiyacını karşılamıyor. İlgili uluslar-arası örgüte ya || ya da TSE; TBD'ye "değişiklik ihtiyacı (request for change)" yazısını da hazırlayalım"dır. Önerilen Sistem Çözümleme Vizyonu budur.

Bölümde verilen başlangıç düzeyi derslerin öğretim üyelerine, aşağıda verilen bilişim alanındaki önemli ISO standarından sözetme mesajı verilmiş olmasına karşılık, yine de öğrencilerin bunları çok önemsemedikleri, hatırlamadıkları farkedilmiştir. IKU-CSE042 Software Engineering dersi (SWE-course) içinde on– oniki ISO, IEEE-std ve TS5881 standardı hatırlatılmakta, öğrenciler öğrenmeye yönlendirilmektedir. Her yazılım Kulubesine eklenmiş Yazılım Kalite Kontrol elemanının en önemli görevi, gerçekleştirdikleri uygulama alanındaki uluslararası standardları gözden kaçırmamaktır.

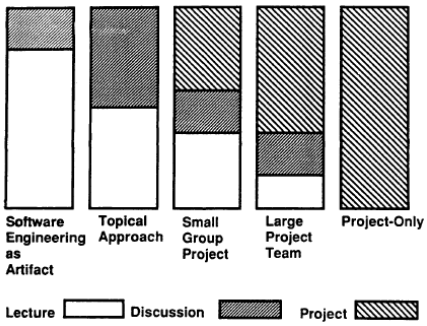
IKU-CSE042 Software Engineering dersi (SWE-course),"Yazılım Mühendisliğine Giriş" temel kavramları, Sommerville[9] yanında, Pfleeger – Atlee[10], vanVliet[11], Boehm[20] den alınan daha somut örneklerle desteklenmektedir. Hatta, günümüzde unutulmuş gibi görünen yazılım mühendisliği alanındaki temel kuramlardan biri olan "Yazılım Fiziği (Software Physics)" [12] için de 2-3 ders saati ayrılmaktadır.

3. Yazılım Kulubesi ve Yazılım Kalite Kontrol Elemanı

On yıllardır kullanılmış ve kullanılmaya devam eden "Yazılım Kulubeleri (Software Hut)"[7] yaklaşımı öğrencilere takım halinde yazılım geliştirme alışkanlığı kazandırmaktadır. Bu alanda kavamlar karıştığında, Shaw & Tomayko yazılım mühendisliği öğretim yaklaşımını[8] beş altkümeye ayırarak sınıflamıştır

Yazılım Geliştirmeyi, küçük sayıda (4 – 8 öğrenci) öğrenci ile öğretmek, bilişim ve bilgisayar literatüründe çok tartışılmıştır. Bilişim ve bilgisayar lisans öğretiminde 1990 öncesini özümseyip, 1991 sonrasına rota çizen en önemli yayın olan IEEE-CS & ACM ccCS1991[8] raporunda da yazılım Kulubelerine referans verilmiştir.

Bizim, IKU-CSE042 Software Engineering Dersinde kullandığımız model aşağıdaki şekildeki, ortadaki kuramsal ders ile laboratuar yükünün dengeli olduğu yapıdır (Şekil 1). Bu yapı, haftada iki kuramsal *(akademik yarıyıl içinde 28 – 30 saat)* ve iki laboratuar *(akademik yarıyıl içinde 28 – 30 saat)* saatinden oluşmaktadır.



Şekil-1: Yazılım Müh. dersi uygulama modelleri [8].

Birinci yazar yönetiminde, çeşitli Türkiye üniversitelerinde *(2001 öncesinde Hacettepe, 2001-2003 Bahçeşehir, 2006-2007 Başkent, 2007 – 2010 İstanbul Kültür-ikinci yazar ile birlikte)* "Yazılım Kulubesi (Software Hut)" yöntemi uygulanmış ve öğrenci grubu içine bir "Yazılım Kalite Kontrol Elemanı (Software Quality Control Member)" eklenmiştir. Bu öğrencinin "yazılım geliştirme sorumluluğu olmadığı"nın laboratuar kuralları arasında belirtimi her yıl tekrarlanmaktadır.

Yazılım Mühendisliği dersi'nde "yazılım kulubesi kuruluş algoritması" ve "yazılım kulubesinde roller" aşağıda iki çerçevede özetlenmiştir:

adım01: Öğretim üyesi öğrenci kayıtlarının son günü, son saatlerde derse kaydolan öğrencileri Genel Not Ortalaması (GPA) yönünden büyükten küçüğe doğru sıralar

adım02: Öğretimüyesi kayıtlı öğrencilerin en yüksek GPA'ya sahip beşte birini (ilk %20) "Yazılım Kalite Kontrol Elemanı" olarak etiketler,

adım03: Öğretim üyesi öğrencilerin en yüksek GPA'lı ikinci beşte birini (ikinci %20) "Yazılım Üretim Yöneticisi" olarak etiketler,

adım04: "Yazılım Kalite Kontrol Elemanı" ve "Yazılım Üretim Yöneticisi" listeleri yarıyılın açıldığı Pazartesi günü 9.00 da ilan edilir.

adım05: Bir "yazılım kulubesi (Software Hut)" kurmak için anlaşan bir kalite kotrol elemanı ve bir üretim yöneticisi, üç gün içinde kendilerine

bir proje tanımlar *(beklenti 4.0 KDSI den büyük olmalı)* ve kulubelerine bir renk almak için dersin asistanına başvurur.

adım06: Yazılım kulubesi, yapılacak proje belli iken, derse kayıtlı öğrencilerin geri kalan %60 I içinden iyi çalışacağına inandıkları üç kişiyi "programcı" olarak işe alırlar.

adım07: Geri kalan onüç hafta içinde, onüç aylık bir iş anlaşmasına benzer bir süreç uygulanır. Öteki deyişle, çalışmayan, işe gelmeyen *(laboratuar saatinde üyesi olduğu yazılım kulubesi ile birlikte çalışmayan)* eleman bir yazılı uyarı verildikten sonra işten (yazılım kulubesinden) çıkarılabilir *// çıkarılır.*

adım08: Sunum: yarıyılın son haftasında, geliştirilen yazılımın uyduğu "uluslararası standartlar", "makro tasarım kararları", "mikro tasarım kararları" listesinin kalite kontrol elemanı tarafından, sağlanmış "işlevsel özellikler", ekran görüntüleri ve "yazılım üretkenliği" bilgilerinin üretim yöneticisi tarafından yapıldığı diğer yazılım Kulubeleri önünde bir sunum yapılır. Diğer yazılım Kulubeleri, uluslararası standarda uymayan yaklaşım ya da çalışmayan işlev bulursa puan alır.

adım09: Yazılım kalite kontrol elemanı, son hafta (sunum) öncesinde grubunu standard hataları, korumalı program hataları vb yanlışlardan korumak için içe dönük olarak çalışır. Ancak, son haftada, bulunamamış hata var ise bunları dersin asistanına rapor eder. *(dersin asistanı, "alıcının kontrol mühendisi rolünü" oynar)*.

adım10: Yarıyılın en başarılı üç yazılım kulubesinin hatıra fotoğrafı dersin öğretim üyesi/asistanı tarafından çekilir. Mezuniyet yıllığı yayınlanacaksa resim orada kullanılacağı gibi, birkaç hafta bölüm bilgi sayfasında yayınlanabilir.

Şekil-2: Yazılım kulubesi kuruluş ve kapanış algoritması

Horning ve Wortman'ın 1977 da yazmış olduğu bildiride [7], "yazılım kulubesi" içinde bir Yazılım Kalite kontrol elemanı bulunması gerektiğini öngörmemiştir.

Alıcı rolü : Bu rol dersin öğretim üyesi tarafından oynanır. Ancak, bu pasif bir rol değildir. Öğretim üyesi aynı zamanda tüm laboratuarlara girer ve gerek ilgili alandaki standardlar ve HCI yeterliliği, kullanışlılığı konusunda destek olur.

Alıcının kontrol mühendisi rolü : Bu rol dersin asistanı tarafından oynanır. Ancak, bu pasif bir rol değildir, dersin asistanı aynı zamanda tüm laboratuarlara girer ve gerek "korumalı yazılım (defensive software), gerekse istemci-sunucu ve internet ortamındaki bağlantılar, bağlantı güvenliği, veri tabanı vb konusunda destek olur. Aynı zamanda iki haftada bir kısmi degerlendirme yapar. Kolay belgele yöntemine dayanarak program boyu kestirimlerini biriktirir.

Yazılım Kalite Kontrol elemanı : İşin programlama yükü olarak paylaşımında doğrudan klavyeye oturmak için görev almaz. Ancak daima grubun içindedir. En kritik modülü yazan üretim yöneticisinin ve programcıların kaynak programını okuduğu, veri giriş korumaları, belgeleme yüzdesi, program içi belgeleme anlaşılırlığı, program yazma standardlarına uyum (camel-Case kullanımı vb) gibi konularda daima aktif olduğu için "eşli programlama (extreme programming)" yöntemine yakın bir rol sayılabilir

Yazılım Üretim Yöneticisi : Bu rol, bölümün mezuniyet adaylarının, yüksek not ortalamasın-da ikinci %20 si tarafından yüklenilen bir roldür. Grubun, üniversite dışında, hafta sonla-rında çalışma motivasyonunun oluşturulmasında liderlik yapması da beklenmektedir.

Şekil-3: Yazılım kulubesi yönteminde roller

"Yazılım Kalite Kontrol Elemanı" kavramının 1991 sonrasında yazılım klubesinde başka üniversiteler tarafından kullanılmış olup olmadığı konusu Ocak 2010 içinde, akademik kaynaklar *(İKÜ'nün abone olduğu akademik veri tabanları)*, googleScholar ve copernic yazılımı ile tekrar araştırdık. Bulgularımız tersine kronolojik sıra ile şunlardır:

2009 Sheffield : Geçen yıl Chris Thompson ve Mike Holcombe (Univ. of Sheffield) son altı yıldır Sheffield Üniversitesinde başarılı biçimde "Yazılım Kulubeleri" yöntemini kullanmış 73 projenin bilgilerini bir teknik not [13] olarak derlemişlerdir. Yöntem, bölümün diğer öğretim üyelerince de kullanılmakta ve sonuçlar çeşitli toplantılarda sunulmaktadır [14].

2001 the Catholic Univ of USA ve Sheffield: Holcombe ve ekibi, Sheffield üniversitesinde, "Crossover Project", "Software Hut" yapısına, ayrıca dördüncü sınıf öğrencilerinin kurduğu ve Genesys adını alan, gerçek projeleri ve gerçek müşterileri olan bir firmaya "eşli programlama (Extreme Programming)" öğrettiklerini açıklamaktadır [15]. Aynı yıl Twedt, Teseriero, Gary, Washington DC deki ABD katolik üniveritesindeki sekiz derslik "yazılım fabrikası

(Software Factory)" ders disisinden söz etmektedirler. Her yarıyıla bir yazılım üreten ders [16] yerleştirmiş olan bu üniversitenin programı, her yarıyılında bir "yazılım lab" bulunan Hacettepe Bilgisayar programına benzerlikler göstermektedir

1997 Lougborough, Derby, Durham : Hem iki farklı üniversitede, hem GPT yazılım firmasında çalışan Ray Dawson – Ron Newsham [17], bir firma (proje dersi düresi 74 saat, yazılım Kulu-besi eleman sayısı 3 – 5, uygulama süresi iki hafta) ve iki üniversite (proje dersi süresi 80 saat, yazılım Kulubesi eleman sayısı 4 – 8, uygulama süresi on hafta) yöntemin kullanılışını açıklamak-ta, GPT yazılım firmasında, aniden elektrik ke-sintisi, gereksinme tanımı değiştirme ve benzeri konularda daha dinamik ve baskılı bir eğitim kullandıklarını belirtmektedir. Aynı yıl Drum-mond ve Boldyref'in university of Durham'da da yöntemi kullandığı gözlemlenmiştir.

1996 Sussex: Sussex üniversitesinde 1995 yılında yöntemi kullanan Steve Easterbrook ve Theodoros Arvanidis, yöntemin özgün biçiminde bulunan Kulubelerin birbirine yazılım bölümü satışı üzerinde durmuşlardır.

1994-1977 Sheffield: "Yazılım kulubesi" yönteminin kullanılması öncesinde İngiltere'de Sheffield Üniversitesi, "Crossover Project" adını verdikleri, öğretim üyesinin daha aktif olduğu bir yazılım geliştirme proje dersini tanımlamışlardır (yöntemin izini 1994 e kadar geriye doğru Holcombe'nin yazılarında izledik, Holcombe 1994 deki yazısında, yazılım Kulubelerinin onbeş yıldır kullanıldığını belirtmektedir).

Yukarıdaki referanslarda görüldüğü gibi, "Yazılım Kulubesi (Software Hut)" yaklaşımı, biraz da oyun gibi ortamda çeşitli üniversiteler tarafından 1977 den 2009 a kullanılmış bir yöntemdir.

Bu yayınlarda, grubun içine bir "yazılım kalite kontrol elemanı" yerleştirme, kavramı bizim ulaşabildiğimiz ve kısmen yukarıda özetlediğimiz yayınlarda yer almamıştır. Bu yaklaşım, birinci yazarın dört Türkiye üniversitesinde uygulamış olduğu özgün katkı olabilir.

4. Hedeflerin Ön Tanımı

Her yarıyıl başında, CSE042 Yazılım Mühendisliği dersinin ilk laboratuarında, YKGS08 açılış konuşmasına [19] dayanarak hedef olarak "yazılım kalite kontrol elemanı da sayıma dahil olmak üzere, birey başına 0.8 KDSI yı aşan yazılım Üretkenliği ve 0.34 hata / KDSI amacı" belirtilerek, bu amaç ve heyecanla (motivation) laboratuar çalışmalarına başlanmaktadır. Profesyonel yazılım geliştirmede 0.34 hata / KDSI çok yüksek konulmuş bir hedef değildir. Kritik sistemlerde hata oranı bu değerden çok daha düşük olmalıdır. Ancak yine de 3,000 program satırında, sadece bir hatanın bizi bu sınıra getireceği dikkate alınmalıdır.

5. Yazılım Laboratuarındaki Ölçü Parametreleri ve Sonuçlar Nedir?

Yazılım Mühendisliği kaynakçasında (literatür) bulunan ve bazıları da bu derse özel olarak tanımlanan parametre ve sayım biçimleri aşağıda tanımlanmıştır. Son üç yıldır IKU-BM de bu ölçümler yazarlar tarafından yapılmaktadır. İzleyen yıllarda aynı bölgedeki üniversitelerin bölümleri ile değerlendirme yöntem ve süreci paylaşılabilir.

Makro Tasarım Kararları (Macro Design Decisions): Beş öğrencinin ürününden oluşan, yaklaşık 4.0+ KDSI dolayında beklenen, günü-müz yazılım boyutlarına göre "mikro boyutlu bir yazılım" için "Yazılım Mimarisi" sözcüğünün yıpratılmasından kaçınılmıştır. Bunun yerine, yazılımın hangi işletim sistemi, hangi veri tabanı altyapısı ile, hangi yazım dili kullanılarak yazıla-cağını belirten, sonradan değiştirilmesi daha zor olan birkaç karar bu grupta belirtilmiştir. Yazılımın, bazı ekranlarının istemci-sunucu yapısında dağılımı da bu gruptadır.

Mikro Tasarım Kararları (Micro Design Decisions): Veri sözlüklerinin hangi uluslararası standarda göre tanımlanacağı (örneğin ülke kod-larını ISO3166-2009-part 1 kullanılacak, dil kod-ları ISO639-2002-part1 kullanılacak, insan ad ve soyadları ayrı bilgi alanları ile alınacak ve herbiri onbeşer karakter olacak, insanYasi değişkeninin tanım alanı 0 – 125 arası kabul edilecek, bu alan dışındaki veri girişleri reddedilecek gibi...)

Kolay-Belgele: [18] Birkaç Türkiye üniversitesinin ve en az bir yazılım firmasının kullandığı, program bölümlerinin (fonksiyon, nesne, vb) bir satır ile tabloda gösterildiği, dört sayfada örneği ile anlatılabildiği basit bir yöntemdir. Her satırda (fonksiyon, nesne, vb), yazılım biriminin kısa adı, kimin yazdığı, kaç programlama dili satırı olduğu, program içi belgeleme oranı gibi birkaç parametre içerir.

Dönem Sonu Gözlemlenen Hata Sayısı : Dönem sonu raporlarında kalite kontrol elemanı tarafından rapor edilen ve gerçek-zamanlı sınama sırasında gözlemlenen eksikler ile proje teslimiden sonra farkedilen toplam hata miktarını belirtmektedir. Bu hatalar çoktan aza doğru başlıklar halinde listelendiğinde yetiştirilemeyen ya da düzgün çalışmayan alt yordamlar ve modüller, mantık ya da işleyiş hataları *(gerçek hayatta yapması gereken işin dışında bir iş yapılması),* programın çalışması sırasında uyumsuz veri ya da yanlış veri kullanımından dolayı ortaya çıkan hatalar, yanlış veri tabanı tabloları ilişkileri ya da tablolarda eksik veri kullanımından doğan hatalar, şeklinde sıralanabilir. Arayüzlerde ortaya çıkan uyumsuzluk ya da eksiklikler ise hata olarak değil genelde eksiklik olarak değerlendirilmiştir.

Çoklu Dil Desteği : Oluşturulan her projede proje bitimiyle birlikte beklenen arayüzlerin "Türkçe" ile birlikte en az bir farklı dili (İngilizce) ile desteklemesi gerektiği projelerin başlangıcında belirtilmiştir. Yıl sonu notlandırmasında baz alınacak ana konulardan birisi bu olmuştur. Sadece arayüzlerde kullanılan her bir nesnenin yazı (text, caption) bilgileri değil aynı zamanda hata mesajlarının ve bilgi mesajlarının da çoklu dil desteği vermesi gerekmektedir.

Rol Yönetimi : Rol yönetimi proje büyüklüğüne göre proje kalitesini gösteren en önemli göstergelerden biridir. Rollerin oluşturulması sırasında erişim haklarının; veri katmanı, iş katmanı ve arayüz katmanları düzeyinde tanımlanmasının yanında çok katmanlı mimarinin işlev yönetimini kontrol eden üst katmanlarından birisi olarak da düşünülmektedir. Elde edilen projelerde belli başlı rol yönetim örnekleri; web tabanlı projelerde öne çıkan seçim (menu) ya da sayfa nesnesi erişimi, sayfaların erişim durumu ve klasör yetki kontrolleri rol yönetim sistemleri ile oluşturulmuştur. Windows tabanlı projelerde yukarıdaki rol yönetim kullanımlarına ek olarak veri katmanı erişiminde, özellikle veri tabanlarında, kullanıcılar tabloların belirli kısımlarına okuma ve yazma yetkisi verilerek rol yönetimi oluşturulmuştur. Bunun sebebi aynı yerel ağda bulunan bilgisayarlardaki erişim haklarının üst kullanıcı *(superadmin)* gibi bir kullanıcı ile tanımlanması durumunda sözkonusu kullanıcı bilgisi ile tüm veritabanını etkileyecek erişime açık olmasını engellemektir.

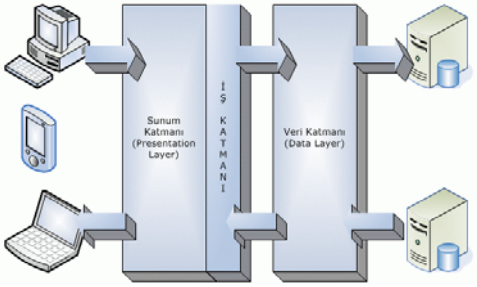
Kullanılan ISO Standardı Sayısı : Her bir grubun kendi projesi ile ilgili ISO standardı projenin kalitesini göstermesi adına bir etken olarak düşünülmüş olup katma değer olarak ele alınmıştır. Bunlardan bazılarını belirmek gerekirse;

- ISO 639-2002 part1; Alpha 2 Language Codes (Doğal Dil kodları Standardı)
- ISO2108:2005; Information and Documentation – International Standard Book Number (ISBN)
- ISO/IEC 3166-1; Alpha-2 code elements: Official Short names and code elements (Ülke Kodları, 2009 içindeki erişimde 246 ülke kodu)
- ISO3297:2007; Information and Documentation – International Standard Serial Number (ISSN)
- ISO 3779:2009; Road Vehicles- Vehicle Identification Number (VIN)- content and structure ayrıca ilgili Yazılım Kulubesi FMVSS115 part 565 (Amerika ve Kanada) ve SAE J853'i de incelemiştir.
- ISO/IEC4217:2008; Codes for representation of currencies and Funds (Para ve fon gösterim kodları)
- ISO 5218:2004-12-01; Information Technology – Codes for representation of human sexes (Cinsiyet Kodları Standardı)
- ISO/IEC 8601: 2002; Information technology- Date and Time formats- Zaman Standardı :
- ISO10087:2006;Small Craft-Craft Identification – Coding system (Gemi/Tekne Gövde Standardı : HULL ID / HIN Standardı)
- ISO/IEC 18033-3:2005 Part 3; Information Technology – security techniques – Encryption Algoritms – Part 3 : Block Ciphers
- ISCO-08 ; International Standards Classification of Occupations (in http://www.ilo.org/ global/ what-we-o/statistics/classifications/lang-- en/index.htm) ve ILO (International Labour Organization), UN (United Nations), ISO (International Standards Organization) müsvette çalışmaları
- tanı kodları, tedavi kodları, ameliyat kodları (http://www.saglik.gov.tr/TR/belgegoster.aspx) ve WHO (World Health Organization), ICD-10 codes

CSE042 Yazılım Mühendisliği dersinin bu bölümü, eğer bölümde dersler güncel standard odaklı olarak verilmemişse bazı öğrenciler için "bir kültür şoku" niteliğinde olabilmektedir. Bazı öğrencilerin ISO, IEEE, IEC, WHO ILO vb örgütlerin adlarını ve standardlarını ilk kez duydukları hissedilebilmektedir. Aynı fiziksel laboratuarı paylaşan 4 – 5

yazılım Kulubesi (20 – 25 öğrenci) ve aynı öğrenci kantinini paylaşan öğrenciler üzerinde çalıştıkları Yazılım Kulubesi ile ilgili buldukları standardları ilgi ve heyecanla farketmekte ve birbirlerine aktarmaktadır.

Katmanlı Mimari : Tüm projelerde beklenen katmanlı mimari yapısında en az iki katmanlı (sunum, veri) yapı oluşturulmuş ise katma değer olarak projelerin değer kazanacağı bilgisi verilmiştir (Şekil 4). Bazı projelerde 3 ve daha fazla katmanlı mimari yapısı kullanılmıştır. Yeni geliştirilen yazılım yapıları ile 4-5 katmanlı (sunum katmanı, güvenlik ve kontrol katmanı, iş katmanı, web servisleri, veri katmanı, veritabanı yönetim katmanı gibi ...) yapıların oluşturulduğu gözlemlenmiştir.



Şekil-4: İki katmanlı mimari yapı

Katmanlı mimarinin sağlamış olduğu ölçeklenebilirlik, kolay yönetilebilirlik, kolay bütünleştirme (entegrasyon), tekrar kullanılabilirlik gibi özel-likleri öğrenci-lerin gerçek hayatta da tasarlayıp, uygulayıp uygulayamadığı gözlemlenmiştir.

6. Sonuç ve Öneriler

"Mezun adayı öğrencilerin ortalama "program üretkenlik düzeyleri" artış göstermiştir (ortalama)

2007-2008: 0,585 KDSI

2008-2009: 0,850 KDSI

2009-2010: 1,040 KDSI

"Alıcıya teslim edilen her bin satır yazılımında hata sayısı (average error per KDSI) düşme eğilimi göstermiştir. (ortalama)

2007-2008: 5,26 Bin satır başına hata (error per KDSI) (18 hata/3,42 KDSI)

2008-2009: 4,40 Bin satır başına hata (error per KDSI) (15,1 hata/3,43 KDSI)

2009-2010: 2,64 Bin satır başına hata (error per KDSI) (9,64 hata /3,65 KDSI)

Ancak, olumluya doğru bir gidiş var ise de, nedenlerinin çözümlemesinde tedbirli olmak ve incelemeye devam etmek gerekir görüşündeyiz.

Olası etkiler şunlar olabilir:

A) 2004 yılında Prof. Dr. Murat Taylı tarafından yapılmış olan tüm dersleri uygulamalı hale getiren yeni IKU-BM ders programı Bu program IEEE-CS & ACM CCCE2004 ile yaklaşık % 90 uyumludur. Programın sistematik olumlu etkisi olduğu düşünülmektedir.

B) Son ikibuçuk yıldır, bölüm başkanlığının çeşitli sınıfların "programlama becerisi çıktıları" açısından hedefler koyması ve bu hedeflerin sosyal ortamda belirtimi (Henüz birinci, ikinci ve üçüncü sınıf sonunda öğrenci becerileri ölçülmemiştir).

C) bu gelişme CSE042 Software Engineering dersinin yerel (kendi içinde) başarı eğilimi de olabilir.".

Hedefler, profesyonel ürün düzeyi kadar yüksek olmamakla birlikte, öğrenci motivasyonunu yükseltmiş ve başarı rotasına sokmuştur.

7. Kaynaklar

[1] Karakas, Ümit M. Mühendislik Üretim Ortamında bir yaşam biçimidir ! Yazılım Mühendisliği Eğitim – Öğretiminde bir Model: Yazılım Firmaları ile Yakın İşbirliği ; UYMS'05, 22-24 Eylül 2005, Ankara, ISBN 975-395-937-0, s. 173- 182

[2] Karakaş, Ü. M.. & Taylı, M. AB uyum sürecinde Bilgisayar Mühendisliği Programlarında gerekli rota değişikliği: Bilgisayar Mühendisliği ve Yazılım Mühendisliği ayırımı; EEB'06: Elektrik, Elektronik, Bilgisayar Mühendislikleri Eğitimi Sempozyumu, İstanbul, EMO 2006.

[3] Bagert, D. J. ve Ardis, M. A. Software Engineering Baccalaurete Programs in the United States : An Overview; 33rd ASEE/IEEE frontiers in education–Conf.; Nov. 5 – 8 Boulder, CO, ISBN 0-7803-7444-4/03.

[4] Karakaş, Ü. M. Yazılım Mühendisliği artık bir üniversite dersi adı değildir ! Yazılım Mühendisliği artık bir Mühendislik Bölümü adıdır. Bilişim Kongresi 2003, İstanbul, Eylül 2003.