

Yazılım Mühendisliği Proje Derslerine Endüstriyel Yaklaşım

Zeynep Altan

Beykent Üniversitesi, Yazılım Mühendisliği Bölümü
zeynepaltan@beykent.edu.tr

Özet: Yazılım Mühendisliği programlarının iki yarıyıllık yazılım tasarımı ve projesi derslerinin amacı, öğrencilerin endüstrinin gerçek dünya problemlerini temel yönleri ile öğrenmeleridir. Yazılım mühendisliği disiplininde “capstone projects” olarak adlandırılan bu derslerde öğrenciler yazılım tasarımı, gerçekleştirilmesi, testi ve yapılandırma yönetimi uygulamalarının etkin olarak nasıl gerçekleştirdiğini öğrenirler. Bu dersler çoğunlukla 3-4 kişilik küçük gruplarla yürütülür; fakat böyle bir çalışma akışı ile öğrenciler ileride çalışacakları şirketlerde karşılaştıkları problemlerin pek çoğuna oldukça sınırlı bir bakış açısından yaklaşmış olacaklardır. Oysa 20-30 öğrencinin birlikte yürüteceği daha büyük projeler daha sistematik bir çözüm verebilir. 100-120 kişilik sınıflarda yürütülecek bu dersler ortalama dört farklı çalışma grubu ile devam ettirilebilir. Her bir grupta kod geliştiriciler ve mimari üzerinde çalışanlar, ürün&satış bölümü analistleri, yapılandırma yöneticileri ya da test uzmanları olarak farklı bir rol üstlenen öğrenciler bulunur.

Fazla sayıda öğrenci içeren çalışma grubunun izlemesinde yapıcı birliktelik (constructive alignment) öğrenme tekniği isteksiz ve ilgisiz öğrencilerin de çalışmaya katılımını sağlayabilir. Amaç öğrenciye bilgiyi yılsonu notu iletmek değil, onları aktif öğrenme içine aktarabilmektir. Bu ise öğrenme çıktılarının, öğrenme aktiviteleri ve öğrenme yaklaşımları ile desteklenmesidir. Böylece öğrenciler SEEK (Software Engineering Education Knowledge) sınıflandırmasındaki özellikleri taşıyan yazılım mühendisleri olarak lisans programlarını tamamlayacaklardır.

Öğrenme çıktısı her bir çalışma grubunun kazanımları olacaktır. Kalabalık öğrenci gruplarını birlikte çalıştırmadaki zorlukların ve olumsuzlukların öğrenciye getireceği yararlılardan fazla olmaması için proje seçimi çok önemlidir. Ayrıca proje konusunun öğrenciler arasında tartışmaya yol açmaması gerekir; bu da isterlerin esnek Ayrıca çevik yöntemlerle çözümlenecek problemlerde öğrencilerin birbirinden farklı roller taşıması da olasıdır.

Etkin bir öğrenme çıktısı elde etmek için, öğretme ve öğrenme aktivitelerinin de dikkatli olarak hazırlanması gerekir. Dersin yürütücülerini öğrenci gruplarının çalışmalarını yönlendiren değil, sadece yol gösteren olarak izlemeli ve bunun için de öğrencilere doğru sorular sorabilmelidir. Bununla birlikte, öğrenme/öğretme sürecinin dikkatli olarak izlenmesi gerekir. Çünkü öğrenciye sağlanan ileri derecede özgürlük amaçlanan öğrenme çıktısını elde etmeyi olanaksızlaştırabilir. Gruplar yenilikçi düşüncelerle geliştirdikleri parçaların birbiri ile hiçbir ilişkisi olmadığı çözümler sunabilirler.

Anahtar Sözcükler: Yazılım Mühendisliği Proje Dersi, Yazılım Tasarımı, Yazılım Testi, Proje Yönetimi, Yapıcı Birliktelik, Öğrenme Modeli, SEEK (Software Engineering Education Knowledge).

1.Giriş

“Software Engineering Programmes are not Computer Science Programmes” isimli makale [1] 90’lı yılların sonlarında yazılım mühendisliği eğitim programlarının bilgisayar bilimleri programlarından farklı bir yaklaşımla hazırlanması gerekliliğini kesin olarak ortaya koymuştur. Çünkü o günlerde bile iş dünyasında çalışmaya başlayan genç mühendisler iyi çözümlenmiş ürünler geliştirememekteydi. Bunun temel nedeni ise öğrencilerin eğitimleri süresince gerçek bir yazılım geliştirme tecrübesi edinemedikleriydi. O dönemde yazılım mühendisliği eğitim programlarının hazırlanması konusundaki bir çalışma, “The Guide to Software Engineering Body of Knowledge– SWEBOK” isimli 1998 yılında başlatılan projedir. Bu proje IEEE tarafından desteklenen yazılım mühendisliği standartlarına odaklanmıştır. Yazılım mühendisliği lisans programlarının sürekli olarak güncellenmesinde, hala devam etmekte olan bu

çalışmaların etkisi büyüktür. Fakat eğitim programları ile ilgili henüz bir fikir birliği sağlanamadığı da bir başka gerçektir. ACM/IEEE-CS Yazılım Mühendisliği programı kılavuzunda [2] gerçek-dünya problemlerinin çözümünde yazılım mühendisliği kavram ve becerilerinin öğrenilmesi odaklı bir program önerilir. Yazılım sistemlerinin günümüzdeki karşılığı, endüstriyel olarak güçlü yazılım ürünlerinin geliştirilmesidir. [3]. SWEBOK kılavuzunun temel yapısını “SEEK- Software Engineering Education Knowledge” sınıflandırmasına uygun bilgi alanları¹ [4] oluşturur. Bu çalışmada bu bilgi alanları da göz önüne alınarak lisans eğitiminin son sınıfında iki yarıyıl süren

¹ SEEK bilgi alanları: Yazılım Gereksinimleri, Yazılımın Oluşturulması, Yazılım Tasarımı, Yazılım Testi, Yazılım Bakımı, Yazılım Yapılanış Yönetimi, Yazılım Mühendisliği Yönetimi, Yazılım Mühendisliği Süreci, Yazılım Müh. Araçları ve Yöntemleri, Yazılımın Niteliği

proje çalışması derslerinin büyük çalışma gruplu projeler olarak gerçekleştirilmesi önerilmektedir.

Broman [5], yazılım mühendisliği proje derslerinin eğitim programlarının omurgası mı yoksa kuyruğu mu olduğu sorusunu sorar ve bunları iki farklı kategoriye ayırır. Bunlardan ilki grup çalışması, sunumlar, biçimsel incelemeler vb. çalışmaların içerildiği her yıl tekrarlanan proje-odaklı sınıflardır. Diğer kategorideki proje çalışması ise öğrencinin eğitim programının başından itibaren öğrendiği bilgi ve becerini uygulamasını amaçlayan ve son yıl programında yer alan geniş kapsamlı bir çalışmayı amaçlayan *capstone project* olarak adlandırılır. İlk kategorideki proje sınıfları eğitim programlarının omurgası olarak tanımlanırken, ikinci kategorideki proje sınıflarının problemlerin çözümünde kuyruk yaklaşımı oluşturduğu öne sürülür. Bu çalışmada eğitim koşullarına uygun yaklaşım olarak ikinci sınıflandırma önerilmektedir.

Büyük proje modelleri ile çalışmanın olumlu yönleri olmasına rağmen, zorlukları da kaçınılmazdır. Yıllardır öğrencilerin büyük projeler üzerinde çalışmalarının önerilmesi [6] ve SEEK sınıflandırmasındaki bilgi alanlarının [7] öğrenci grupları arasında dağıtılarak incelenmesi çalışmaların doğru gerçekleştirilmesi için önemlidir. Fakat çok sayıda öğrencinin birlikte çalışacağı projelerde tek bir sorumlu öğretim üyesi olmayıp çalışmanın bir komisyon tarafından izlenmesi öğrencilerin ferdi olarak doğru değerlendirilmelerinde önemlidir. Fazla sayıda öğrenci içeren çalışma grubunun izlemesinde yapıcı [8] öğrenme teknikleri ilgisiz ve isteksiz öğrencilerin de çalışmaya katılımını sağlayabilir. Amaç öğrenciye bilgiyi not alma ve geçme kaygısı ile iletmek değil, onları aktif öğrenme içine aktarabilmektir.

2. Büyük Proje Gruplarının Güçlükleri ve Çözümü

Yazılım mühendisliği tasarım ve proje derslerinde büyük gruplarla çalışmayı, hem öğretim üyesinin çalışmaları izlemesindeki, hem de çalışmaların değerlendirmesindeki güçlükler engeller. Çünkü öğrencilerin bilgi ve beceri düzeylerinin dağılımı farklıdır. İş dünyasında yöneticiler çalışanlarını subjektif olarak değerlendirir; oysa öğrencilerin değerlendirilmesi birey bazında objektif ve niceliksel, yani ölçülebilir düzeydedir. Ayrıca öğrenciler çalışmalarını para karşılığı değil not karşılığı yaparlar; bu durum öğrencilerin tümünü motive eden bir değerlendirme olmayabilir. Büyük çalışma gruplarındaki olumsuzluğun diğer bir nedeni de öğrencilerin tam-gün çalışma konumunda olmamalarıdır. Etkin bir takım çalışmasını engelleyen projenin tamamlanması için verilmiş olan kesin tarihlerdir; projelerin başında genellikle zamanında tamamlanması güç çalışmaları hedeflenir.

Gayret ve çabalarının tümünü yakın gelecekteki başarısı için yaptığının farkında olan öğrenci sayısı büyük çalışma grubu içinde az olduğu zaman ne olacaktır? Çünkü üretken olmayan öğrenci davranışları çalışmaların nota doğru olarak dönüşümünü zorlaştıracaktır. Not verme tekniğini etkileyen bazı öğrenci tipleri:

“minimum çalışma ile en yüksek notu hedefleyen öğrenci”,

“maksimum çalışma ile en yüksek notu hedefleyen öğrenci”,

“işini yapmayı minimum notu kabullenen öğrenci”,

“sadece başlangıçtaki görev dağılımında çok istekli gibi görünen öğrenci”,

“çalışmanın tüm aşamalarına sadece yüzeysel olarak katılan, işini hiç bir zaman tamamlamayan öğrenci”,

“daha yüksek bir not için işini yapmayanların çalışmalarını tamamlayan öğrenci”,

“daha yüksek not için derecelendirme sistemini istismar eden herhangi bir grubunun elemanları”

şeklinde olabilir[9]. Bu tür durumlarla karşılaşılmasını engellemek için sorumluluk çalışmaların tüm aşamalarında etkin bir izleme gerçekleştirecek öğretim üyesindedir.

2.1 Yapıcı Birliktelik ile Öğrenme

Büyük proje gruplarının çalışmalarında öğretim üyelerinin olası hatalı yönlendirmeleri ve değerlendirmelerinin engellenmesi için *yapıcı birliktelik*²[10] yaklaşımı alternatif bir çözüm olarak önerilmektedir.

Yapıcı birliktelik ile biri öğrenenin bakış açısından, diğer ise öğreticinin performansına göre birbirinden farklı iki görüş öne sürülür. *Yapıcılık* öğrenenin, şimdiye kadar öğrendikleri ve tek başına gerçekleştirebildiği aktivitelerle bilgisini oluşturmasıdır. Öğrenmeye bu açıdan bakıldığında, öğretme öğrenciye bilgiyi iletmek değil, aksine öğrencinin aktif öğrenmesini teşvik etmektir. *Birliktelik* ile *hedeflenen öğrenme çıktıları*, *öğretme ve öğrenme aktiviteleri* ve *değerlendirmeler* olarak tanımlanmış derslerin yürütümü ile ilgili sınıflandırma yapılıdır.

2.1.1 Hedeflenen Öğrenme Çıktıları

Hedeflenen öğrenme çıktıları Tablo 1’de verilmiştir. Öğrencinin proje çalışmalarındaki, yani organizasyon içerisindeki görevi (O) analist, proje yöneticisi, mimariyi yapılandıran, ürün yöneticisi, kod geliştirici ya da test uzmanı olabilir. Öğrenci önceki derslerindeki bir çalışması nedeni ile projenin herhangi bir aşamasında diğer arkadaşlarından daha deneyimli olabilir; bu da projenin o kısmında öğrenciyi lider ya da yönetici yapabilir (S1). Her projede olduğu gibi öğrenci projelerinde de, zamanın planlanması, bütçe

² Günümüzde etkin öğrenme tekniklerinden biri olan yapıcılık (constructivism), öğrencilerin öğretmenleri tarafından iletilen bilgileri sadece almaları ve depolamalarından ziyade bilgiyi yapılandırmalarını (oluşturmalarını) savunur.

(adam-ay), ürünün fonksiyonelliği ve ölçülebilir nitelik faktörleri gibi temel kısıtların göz önüne alınması önemlidir (S2). Büyük öğrenci grupları birbirinden farklı bilgi birikimine ve beceriye sahip arkadaşların iletişimini gerektirdiği için (I1), olası anlaşmazlıkların kısa aralıklarda hazırlanan dokümanlarla (I2) giderilmesi amaçlanmalıdır.

Tablo 1: Hedeflenen Öğrenme Çıktıları

| O | Organizasyonel yapılar ve farklı roller |
|----|--|
| S1 | Önceki ve güncel geliştirme süreçlerinin karşılaştırılması |
| S2 | Karşılaşılabilecek kısıtların analizi |
| I1 | Büyük grupların iletişimindeki güçlükler |
| I2 | İletişimin gerçekleşmesi |

2.1.2 Öğretme/Öğrenme Aktiviteleri

Büyük projelerin etkin olarak yürütülmesi için amaç, öğrencilerin projenin başından sonuna kadar çalışmaları kendilerinin yürütmesi, öğretim üyesinin çalışmalar sırasında yönlendirici olmayıp öğrencilere sadece danışmanlık yapmasıdır. Projenin başında öğrenci projede üstlenmek istediği görevi proje yürütücülerinden birine bir yazı ile önerir. Proje yöneticisi bu kalabalık grup içerisinde gelen önerilere göre görev dağılımını gerçekleştirir. Öğrenciler her hafta tüm grup elemanları ve danışmanları ile toplantılar düzenler. Öğrencilere başlangıçta proje ile ilgili herhangi bir istemde bulunulmaz. İsterler toplantılar sırasında belirlenir ve değiştirilebilir. Proje ile ilgili ön çalışmalar tekrarlanan toplantılarda projenin tamamlanabileceği kesinleştiğinde sona erer. Ön çalışmalar öğrencilerin sınırlı bütçe ile kısıtlı zaman aralığında sorumluluk almalarına, müşteriye ikna edebilmelerine, tercihlerin öncelikli olarak teşvik edilmesini öğrenmelerine yardımcı olur. Çünkü tüm grup elemanlıları tarafından sürekli olarak tekrarlanan bu toplantılar ürünün müşteriye satışının kesinleşmesi amacı ile yapılmaktadır. Toplantılarda müşterilerin grup elemanları dışındaki danışman ve yardımcıları olduğu kabul edilmektedir. Ürünü kabul ettirme, yani satışı ile proje süresi tekrar başa döner ve tekrarlanacak planlamalar ile öğrencilerin ürünü geliştirmeleri teşvik edilir. İterasyon planlama, yani sonraki iterasyonda çözülmesi gereken işlerin neler olduğu ve iterasyon değerlendirme, yani son iterasyonda nelerin gerçekleştirildiği proje çalışması süresince zorunlu olarak yapılmalıdır. Proje süresince tüm öğrencilerin yaptıklarını belirlenen zaman dilimlerinde, örneğin her saat raporlamaları gerekir. Ayrıca proje de başından sonuna kadar tümüyle ile planlanmalıdır. Öğrenciler iş dünyasında olduğu gibi klasik bir organizasyon yapısı oluştururlar: bölüm yöneticileri, proje yöneticileri, test uzmanları ve kod geliştiriciler olmak üzere farklı bölümlere ayrılırlar. Her iterasyonu yapılanların tartışıldığı bir toplantı izler; “olumlu işler”, “kötü giden işler”, “sonraki iterasyonda nelerin, nasıl geliştirilebileceği” gibi konular tartışılır. Projenin sonunda öğrenciler ürünlerini kullanıcıya yönelik olarak, ikna amacı ile

sözlü olarak sunar. Son olarak ta çalışma ile ilgili görüşlerin, ürünün geliştirilmesi sırasında olumlu ve olumsuz tespitlerin belirtildiği ferdi raporlar hazırlanır.

Tablo 2: Öğrenme/Öğretme Aktiviteleri

| | | Öğrenme Çıktısı |
|---|---------------------------------------|-----------------|
| a | Görev seçimi | O |
| b | Toplantılar | S1, S2,I1,I2 |
| c | İsterlerin Belirlenmesi | S1,I1,I2 |
| d | Ön Çalışmalar | S1,S2,I1 |
| e | İterasyon Planlama ve Değerlendirme | S1,I1,I2 |
| f | Proje Planlaması ve Zaman Raporlaması | S2 |
| g | Organizasyonel Yapı | O, S1 |
| h | Geriye Dönük Tartışma | S1,I1 |
| i | Sürümün Planlanması | S1,S2 |
| j | Kişisel Düşünceler ve Dokümantasyon | O,S1,S2,I1,I2 |

Tablo2’de bu adımların tümünün özetlendiği öğrenme/öğretme aktiviteleri ve bunların hedeflenen öğrenme çıktıları ile ilişkisi verilmektedir.

2.1.3 Değerlendirmeler

Büyük öğrenci projelerinde uygulanması önerilen yapıcı birliktelik yaklaşımında son adım değerlendirmelerdir. Bu aşamada amaçlanan öğrenme çıktılarının performansı ölçülür. Performans, çalışmanın süreçlerine ve ürüne bakış açısına göre farklı değerlendirilir. Çalışmanın süreçleri danışmanın katıldığı toplantılardaki gözlemleri, diğer bir ifade ile bu toplantılarda kendisinin yönetici ya da müşteri (kullanıcı) rolü üstlendiğinde öğrencilerle etkileşimlerine göre belirlenir. Ayrıca öğrencilerin her birinin kendilerine atanan görevleri ne oranda gerçekleştirdiklerini kanıtlayacakları raporlar da değerlendirilir. Çalışma süreçlerine göre değerlendirilen öğrenme çıktıları O,S1,S2 ve I1 olarak özetlenir. Diğer değerlendirme ölçümü sonuçta elde edilen ürüne göre gerçekleşecektir. Bunun için ürünün hem nitelik hem de fonksiyonelliğine, müşteri ya da kullanıcı bakış açısından bakılır. Ürüne göre değerlendirilen öğrenme çıktıları S2 ve I2 olacaktır.

2.2 Yapıcı Birlikteliğin Sonuçları

Proje çalışmalarında öğrencilerin çalışmalarını *yapıcılık* olarak genelleştirilen öğrenme yaklaşımı ile gerçekleştirmeleri çalışmaların değerlendirilmesini de güçleştirecektir. Öğrenci odaklı öğrenme modelinde öğrencinin ne öğrendiğini belirlemede, öğretim üyesinin ne yaptığından ziyade öğrencinin ne yaptığı daha önemlidir. Kısaca öğrenmede öğretim üyesi herhangi bir rol üstlenmemektedir. Bu da öğretim üyesinin öğrenci çalışmaları sırasında kullanılacak modeli doğru ve tam uygulamasını gerektirir. Tablo 2’nin sınıflandırılması da bu bakış açısına göre düzenlenmiştir. İteratif öğrenme ile hem yazılım mühendisliğinde tekrarlanarak gerçekleştirilen ürün

geliştirme yöntemleri kullanılmış olacak, hem de öğretim üyesinin öğrenme çıktılarını doğru olarak değerlendirmesi sağlanacaktır. Bu yaklaşımla öğrenci aynı zamanda eğitimi süresince ders programlarında öğrendiklerini de uygulama olanağı bulacaktır.

Öğrenci dönem sonunda değerlendirilirken, birtakım sorularla da ölçümlenebilir. Sorular (kesinlikle) kabul ediyorum ya da (kesinlikle) kabul etmiyorum, yorumsuz, geçersiz gibi farklı seçeneklerle cevaplanabilir. Sorular “herhangi iki notum arasında fark olduğunda, bu durum daha çok çalışmama neden oluyor”, “not kriterleri çalışmamı etkiliyor”, “farklı gruplarla yarışmak beni daha fazla motive ediyor” ve benzeri şekillerde yöneltilir.

Sonuç olarak büyük öğrenci grupları ile yapılan proje çalışmaları öğretim üyelerinin öğrencileri izlemede ve değerlendirmede çok daha fazla titiz olmalarını ve öğrencilerle klasik proje derslerinden çok daha fazla birlikte olmalarını gerektirir.

3. Son Sınıf Proje Gruplarına Alternatif Yaklaşım: Her Yıl Tekrarlanan Proje Grupları

2. Bölümde 7. yarıyıl tasarım projesi, 8. yarıyıl *capstone project* olarak adlandırılan dersleri, yazılım mühendisliği bölümü öğrencilerinin büyük çalışma grupları ile tamamlamaları önerilmektedir. Eğitim programının sonunda alınan mesleki bilgi ve becerilerin pekiştirilmesi için bir fırsat olan bu derslere farklı bir yaklaşım, proje odaklı sınıfların oluşturulmasıdır. Bu seçimde öğrenciler yine gruplar oluşturur, araştırmalar yapar ve bunları dönem sonunda sunarlar. Proje dersleri her yıl açılmaktadır ve programa kayıtlı öğrenciler hangi yılın öğrencisi olduklarına bakılmaksızın karma olarak proje gruplarını oluşturmaktadır. Bu yaklaşımı, 1. Bölümde de ifade edildiği gibi, Broman eğitim programının omurgası olarak değerlendirmiştir.

Eğitim programının son yılında önerilen büyük çalışma grubu yaklaşımında olduğu gibi, bu yeni yaklaşımda da ortalama 30 öğrencili gruplarla geniş kapsamlı projelerin endüstriyel bir yaklaşımla geliştirilmesi mümkündür. Her iki yaklaşımın birbirinden temel farkı, proje dersinin her yıl açılması ile küçük sınıflardakilerin kendilerinden daha deneyimli öğrencilerden yararlanabilmesi, üst sınıflardaki öğrencilerin de belirli konuları derinlemesine anlayarak kendilerini daha iyi geliştirebilmeleridir. Aslında farklı sınıflardaki öğrenciler arasındaki etkileşim yaratıcı bir öğrenme ortamı oluşturmaktadır. Proje konularının endüstriden gerçek müşteriler tarafından belirlenmesi de mümkün olabilir. Yazılım şirketleri çalıştıkları yeni proje konularını öğrencilerle paylaşır. Onların da kendileri ile eş zamanlı fakat bağımsız olarak aynı proje üzerinde çalışmalarını isteyebilir. Böylece öğrencilerin gerçek isterler üzerinde çalışmaları sağlanır. Bu tür

yaklaşımlar hem öğrenciler, hem de çalışma konularını öğrencilerle paylaşan şirketler için yararlıdır. Öğrenciler gerçek dünya problemlerine biraz daha yaklaşırken, yazılım şirketleri ileride daha deneyimli yeni mühendislerle çalışabileceklerdir. Aynı zamanda şirketlerin öğrencilerden bazılarının özgün fikirlerine ve çözümlerine karşılıksız ortak olmaları da mümkün olacaktır.

Öğrenciler proje dersini birden fazla defa almaya teşvik edilerek, çalışmalarının farklı yıllarında farklı roller üstlenme olanağı bulurlar. Böylece de mezuniyetlerinden önce deneyim yoğunluklu ileri düzeyde bilgi ve beceri edinirler.

4.Sonuç

Yazılım mühendisliği bölümlerinin eğitim programlarının profesyonel bir iş hayatı için pek çok eksiklikleri olduğu farklı iş ortamlarının ortak şikâyetleridir. Pek çok yeni mezun çok yönlü becerilerindeki eksikliklerden oldukça fazla etkilenir. Örneğin teknik olarak güçlü genç bir yazılım mühendisinin çevresi ile iletişimde ya da yönetsel becerilerinde sorunlar olabilir; şirketler çalıştırdıkları elemanlarında bunun tam tersi bir durumla da karşılaşabilir.

Bu çalışmada koşulların uygun olması ölçüsünde endüstriden gerçek müşterilerin araştırma konularını öğrencilerle paylaştığı, bu temin edilemediğinde danışmanların müşteri rolünü üstlenerek gerçek dünya problemlerinin çözümlendiği proje dersleri önerilmektedir. Bu derslerin iş hayatına hazırlanmak üzere büyük proje grupları ile yürütülmesinin nedenleri ve nasıl gerçekleştirilebileceği anlatılmaktadır. Bir yıl devam eden son sınıf proje çalışmalarında endüstriyel olarak gerçekçi bir ürünün elde edilmesini maksimum ölçütlerde sağlamak temel hedefdir. Bunun için odaklanılması gereken noktalar organizasyonel yapının sağlam kurulduğu, ürünün geliştirme süreçlerinin doğru belirlendiği ve grup elemanlarının birbirleri ile iletişiminin eksiksiz olarak sağlandığı bir yapılandırma. Böyle bir yaklaşım öğrencilere iş hayatı ile ilgili oldukça yararlı deneyimler kazandırmasının yanında, bir yıllık çalışmanın sonunda nitelik ölçütlerini kısmen de olsa sağlayacak çalışmalar yapılabilirliği öngörür.

Kaynakça

[1] Parnas, D.L., “Software Engineering Programmes are not computer science programmes”, *Annals of Software Engineering*,6,19-37, 1999

[2] Joint Task Force on Computing Curricula, *Software Engineering 2004- Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*, IEEE Computer Society and ACM, 2004.

- [3] Jalote, P. , A Concise Introduction to Software Engineering, 2008 . page 2. Springer.
- [4] Guide to Software Engineering Body of Knowledge 2004 SWEBOK, IEEE Press
- [5] Broman, D., “Should Software Engineering Projects be the Backbone or Tail of Computing Curricula?”, 23rd IEEE Conference Software Engineering Education and Training, 2010.
- [6] Shaw, M., Software Engineering Education: A roadmap. In A. Finkelstein (Ed.), The Future of Software Engineering (pp. 371-380). New York, NY: ACM Press, 2000.
- [7] Altan, Z., Beykent Üniversitesi Yazılım Mühendisliği Lisans Programı, Akademik Bilişim 2010
- [8] Ben-Ari, M., Constructivism in Computer Science Education, 29. SIGCSE Technical Symposium on Computer Science Education, , 1998.
- [9] David, Coppit , Implementing Large Projects in Software Engineering Courses, Computer Science Education Vol. 16, No. 1, pp. 53 – 73, 2006
- [10] Broman, D., Sandahl, K., Abu Baker, M., “The Company Approach to Software Engineering Courses”, Submitted to IEEE Transactions on Education, Preprint 2011.