YAZILIM PROJELERİNDE YAZILIM GELİŞTİRİCİLERİN YAZILIM SÜREÇ MODELLERİNİ KULLANIM FARKINDALIKLARI

Geliş Tarihi (ReceievedDate) 03.07.2018
Kabul Tarihi (Accepted Date) 20.11.2018

Mustafa KESKİNKILIÇ[1]
Esra ÖZMEN[2]

Özet

Yazılım projelerinin vazgeçilmez elemanları yazılım geliştiricilerdir. Yazılım geliştiriciler işlerini yaparken bir dizi süreç, araç ve yöntemler kullanmak durumundadırlar. Kaliteli ve verimli yazılım geliştirmek için sadece en uygun yazılım süreç modelinin, yazılım geliştirme araç ve yönteminin kullanılması bir zorunluluktur. Bir yazılım projesinde kullanılacak yazılım süreç modelinin, sonradan da izlenebilecek kendine özgü adımları olduğu için, proje aşamasından sonra ortaya çıkabilecek hata veya eksiklikler yazılım geliştiricilerden bağımsız olarak değerlendirilebilecektir. Bu yüzden bir yazılım projesinde kullanılacak yazılım süreç modeli, geliştirme araç ve yöntemi yazılım geliştiriciler tarafından kapsamlı bir şekilde bilinmelidir. "Yazılım geliştiriciler yazılım süreç modellerini kullanıyorlar mı?", "Yazılım geliştiriciler yazılım süreç modelleri hakkında bilgi düzeyleri nedir?" sorularının yanıtı, projenin başarısı için önem arz etmektedir. Bu çalışmada, yazılım geliştiricilerin yazılım süreç modellerini kullanım durumları ile yazılım süreç modelleri farkındalık düzeyleri arasında farklılık araştırılmıştır. Ayrıca, yazılım geliştiricilerin yazılım mühendisliği yöntembilimleri hakkındaki algıları, bilgi düzeyleri ve kullandıkları yazılım süreç modelleri hakkındaki farkındalık düzeyleri ortaya konulmaya çalışılmıştır.

Anahtar Kelimeler: Yazılım Projesi, Yazılım Geliştirme, Yazılım Süreç Modelleri

SOFTWARE DEVELOPERS' AWARENESS OF USING SOFTWARE PROCESS MODELS IN SOFTWARE PROJECTS

Abstract

Software developers are the indispensable elements of software projects. Software developers should use a number of processes, tools and methods to do their jobs. It is important to use the most appropriate software process model, development tool and method to improve quality and efficiency software. Since the software developments models are not usually selected by the software developers, they can be held responsible for the model based issues that arise in the projects. Therefore, software process model to be used in a software project, development tools and methods should be known extensively by software developers. Answers to the questions like "Do software developers use software process models?", "What is the level of knowledge of software developers about software process models?" are important for the success of the project. In this study, the differences between software developers' use of software process models and their levels of awareness regarding these models are investigated. Moreover, perceptions of the software developers about the software engineering methodologies, their levels of knowledge, and their levels of awareness regarding the software process models are analyzed.

Keywords: Software Project, Software Development, Software Process Models

1. GİRİŞ

Yazılım Projelerini gerçekleştiren ve bu sürecin vazgeçilmezi olan yazılım sektörünün önemi günümüzde daha iyi anlaşılmaktadır. Günlük hayatın iş süreçlerini elektronik ortamda gerçekleştirmek için birçok yazılım projesi gerçekleştirilmektedir. Gerçekleştirilen bu projelerin kaliteli sonuçlar üretebilmeleri için, proje geliştirilirken yazılım proje yönetimi ile ilgili yöntemlerin kullanılması beklenmektedir (Gürbüz, 2010: 176).

Yazılım, bir bilgisayar sisteminin çalışması, yönetilmesi veya bilgisayarda bir işin yapılması amacı ile oluşturulan programlardır. Bir diğer tanımda ise yazılım, bilgisayar sisteminin çalışmasını sağlayan bilgisayar programları, prosedürler ve ilişkili verilerin bütünü olarak tanımlanır. Günümüzde kısaca problem ile çözüm arasında bir ara yüz olarak değerlendirilebilir.

Bilgisayar yazılımları genellikle sistem yazılımları ve uygulama yazılımları olmak üzere iki sınıfa ayrılmaktadır. Aygıt sürücüleri, işletim sistemleri, uygulama sunucuları gibi sistem yazılımları,

[1] Dr. Öğr. Üyesi, Atatürk Üniversitesi İktisadi ve İdari Bilimler Fakültesi Yönetim Bilişim Sistemleri Bölümü, muskes@atauni.edu.tr
[2] Doktora öğrencisi, Atatürk Üniversitesi Sosyal Bilimler Enstitüsü Yönetim Bilişim Sistemleri Anabilim Dalı, ozmenesra25@gmail.com

bilgisayarın çalışması için gerekli olan ana fonksiyonları yerine getirirken; uygulama yazılımları ise bir arayüz vasıtasıyla kullanıcıdan veri alıp işleyen ve sonuçları yine kullanıcıya gönderen, diğer bir deyişle bilgisayarı kullanıcıların işlerine çözüm bulmalarını sağlayan stok takip programı, personel izleme sistemi, üniversite otomasyonu, muhasebe programları gibi yazılımlardır (Erten, 2016). Ancak bu yazılımların geliştirilmesi işinde tıpkı diğer tüm alanlarda olduğu gibi belirli bir disiplin içerisinde, belirli yöntem, araç ve teknikler kullanılarak yapılması gerekmektedir. Bu noktada yazılımın geliştirilmesi, işletilmesi ve bakım sürecini sistematik, disiplinli ve niceliksel olarak yöneten yazılım mühendisliği disiplini devreye girmektedir (Radatz, Geraci ve Katki, 1990: 84). Yazılım mühendisliği, yazılım geliştirme işinin çeşitli mühendislik yöntemleriyle yapılmasını ileri süren bir disiplindir. Yazılım mühendisliğinin amacı, kaliteli, sağlam, güvenilir ve isteklere uygun yazılım ürünlerinin geliştirilmesini sağlamaktır (Sarıdoğan, 2017). Bu bağlamda yazılım mühendisliği; süreç, araç ve yöntemlerden oluşmaktadır (Pressman, 2010).

Dünyada ve ülkemizde birçok yazılım projesi başlatılmakta ancak birçoğu başarısızlıkla sonuçlanmaktadır (CHAOS Report, 2009). Yazılım projelerindeki temel başarısızlık nedenleri şöyle sıralanabilir.

Yazılım projelerinde zincirleme hatalara ve sonuçta başarısızlığa neden olan ta en başta yanlış proje tanımı yapmaktır. Bu hata yanlış isterler analizine neden olmakta, yanlış isterler analizi (sistem analizi) beraberinde hatalı sistem tasarımını getirmektedir. Bu da hatalı yazılım süreçleri oluşturmaya neden olmaktadır. Hatalı yazılım süreçleri ise yanlış araç ve yöntem kullanımını getirmekte bu da uygun niteliklere sahip olmayan yazılım geliştirici seçimini doğurmaktadır. Tüm bunların sonucunda doğru ve uygun yazılım modelinin de seçimi yapılamayacağından yazılım projesinin başarısız olması kaçınılmaz olmaktadır.

En genel bakış açısıyla, yazılım projelerinin başarısı, yani istenen kalitede yazılım ürünü elde etmenin yolu, süreç iyileştirmeye dayalı yazılım kalite yaklaşımından geçmektedir. Buna göre en başta bir yazılım projesi süreç iyileştirmeyi esas alarak başlamalı; doğru sistem analizi ve tasarımı çerçevesinde yazılım süreçlerini iyileştirmeli; yazılımı geliştirirken uygun araç ve yöntemler ile uygun yazılım süreç modelini kullanarak istenen kalitede yazılım ürünü elde etmeye çalışmalıdır.

Bir yazılım projesinde yazılım geliştirmeyi sağlayan birbiriyle ilişkili ve tutarlı etkinliklerin tamamı yazılım sürecini oluşturmaktadır. Yazılım süreç iyileştirme ise iyileştirme etkinliklerinin planlanması ve uygulanması için gereken iş dizisi, araç ve tekniklerinden oluşur (Aysolmaz, Yıldız ve Demirörs, 2011).

Yazılım geliştirme işi tekrarlanabilir adımlardan oluşan bir süreçten oluşmaktadır. Bu süreci gerçekleştirebilmek için birçok yöntem bulunmaktadır. Hangi yöntem kullanılırsa kullanılsın genel olarak yazılım geliştirme işi, tanımlama, geliştirme ve bakım adımlarından oluşmaktadır. Yazılım geliştirme işleminde belirli yöntemlerin kullanılması, yazılım geliştirirken yapılması gereken aşamaların hatırlanmasını sağlamakta, böylelikle önemli aşamaların gözden kaçmasına mani olmaktadır (Sarıdoğan, 2017). Yazılım süreç modelleri genellikle yazılım evrimini gerçekleştirmeye yönelik stratejileri içeren faaliyetlerin, nesnelerin, dönüşümlerin ve etkinliklerin bağlantılı bir dizisini temsil eder. Bu tür modeller, yazılım yaşam döngüsü faaliyetlerinin daha kesin ve resmileştirilmiş açıklamalarını geliştirmek için kullanılabilir (Scacchi, 2002).

Stephens (2015) yazılım süreç modellerini; tahmine dayalı (precdive) modeller, yinelemeli (iterative) modeller ve hızlı uygulama geliştirme modelleri (RAD) olmak üzere üç başlık altında sınıflandırmaktadır. Bu çalışmada ele alınmış modeller; kod eksenli yazılım geliştirme modeli, doğrusal modeller (şelale modeli – V modeli), yinelemeli geliştirme modelleri (artımlı geliştirme modeli (literatürde artırımlı ve artırımsal olarak da geçmektedir), evrimsel geliştirme modeli, spiral model), hızlı uygulama geliştirme modeli (RAD), rasyonel birleştirilmiş süreç modeli (RUP) ve prototip modelleridir.

Kodla ve düzelt olarak da adlandırılan kod eksenli yazılım geliştirme modelinde ihtiyaç analizi tam olarak bitmeden ya da kısmen tamamlanmış olarak, herhangi bir tasarım yapmadan doğrudan kodlama işlemine geçilmektedir (Nizam, 2015).

Doğrusal modeller şelale ve V modeli olmak üzere iki sınıfa ayrılmaktadır. Klasik çevrim modeli olarak da adlandırılan şelale modeli genellikle gereksinimleri tam olarak belirlenmiş ve gereksinimleri değişmeyen projelerde kullanılmaktadır. Şelale modeli analiz, tasarım, kodlama, test ve destek adımlarından oluşmaktadır (Pressman, 2010). Şelale modelinde ilk olarak gereksinimler belirlenerek gözden geçirilir ve onaylama işlemi yapılır. Daha sonraki adım da yine gözden geçirilerek onaylanır. Sonraki adımlarda da bu şekilde "gözden geçir-uygunsa onayla" işlemi devam etmektedir. Genel olarak şelale modelinde bir adım kendisinden önceki adım tamamlanmadan başlamamaktadır. Örneğin; kodlama adımı, kendisinden önce yer alan tasarım adımı tamamlanmadan başlayamaz. Kodlama adımına geçildiği an tasarım adımının tamamının bittiği anlamına gelmektedir. Şelale modelinin odaklandığı nokta her adımın dokümanlarla gerçekleştirilmesidir (Pauca, 2003). Bu model dokümantasyon odaklı olduğu için hem çok maliyetli hem de uzun zaman almaktadır. Maliyet nedeniyle son aşamaya bırakılan problemler müşteri ihtiyaçlarına dönüt vermeyi zorlaştırmaktadır (Gül, 2006). Müşteri ihtiyaçlarının net olarak anlaşıldığı ve büyük projelerde kullanılması önerilmektedir (Sommervile, 2000). Şelale modelinin en büyük dezavantajı ise değişikliklere karşı esnek olmamasıdır (Sommerville, 2011). Şelale modeli, yazılım geliştirme sürecinde "tasarımdan önce tanımla, kodlamadan önce tasarla" ilkelerini desteklemektedir (Munassar N.M.A., Govardhan A., 2012). V modeli, şelale modelinin biraz daha gelişmiş sürümü olarak adlandırılabilir. Kontrol adımları daha iyi organize edilmiştir. Modelin diyagram gösterimi V harfine benzediği için bu adı almıştır. V modeli test planlarının yaşam döngüsünün daha erken safhasında geliştirilmesi nedeniyle şelale modelinden daha başarılı bir model olarak görülmektedir (Munassar ve Govardhan, 2012; Nizam, 2015).

Artımlı, evrimsel ve spiral yazılım geliştirme modelleri yinelemeli geliştirme modellerindendir. Artımlı geliştirme modelinde ihtiyaçlar başlangıçta büyük oranda bellidir. Modelde sistem fonksiyonlara ayrılmaktadır. Yani müşteri ürünü kullanmak için tüm sistemin bitmesini beklemez ve belirlenen kritik ihtiyaçlarla geliştirilen yazılım kullanılmaya başlanır. Hazırlanan prototiplerle müşteri deneyim kazanır ve süreç içerisinde müşteriye daha çok yer verilir. Şelale modeline göre daha az risklidir ve herhangi bir başarısızlık tüm projeye yayılmaz. En başta kritik ihtiyaçlar belirlendiği için sistemdeki önemli parçalar en çok teste tabi tutulur ve bu parçalarda hata ile karşılaşılma olasılığı oldukça düşüktür (Sommervile, 2000). Evrimsel geliştirme modeli, artımlı geliştirme modelinin biraz daha gelişmiş halidir. Artımlı geliştirme modelinde ihtiyaçların başlangıçta belirli olma gereksinimi bu modelde kaldırılmıştır. İhtiyaçlar tam olarak belli olmasa bile bu model ile yazılım geliştirilebilmektedir (Nizam, 2015). Spiral model, şelale modelinin, artımlı geliştirme modelinin ve evrimsel geliştirme modelinin gelişmiş halidir. Boehm tarafından 1988 yılında geliştirilen modelin her döngüsünde müşteri iletişimi, planlama, risk analizi, mühendislik, kurulum ve müşteri değerlendirmesi gibi görev alanları tekrar edilerek yazılım modeli tamamlanmaktadır. Spiral modelin risk odaklı olması ve sürecin sarmal şeklinde ilerlemesi diğer modellerden farkını ortaya koymaktadır (Sommerville, 2011). Diğer yinelemeli geliştirme modelleri özelliklerine ek olarak risk değerlendirmesi yapmaktadır (Boehm, 1988: 61). Geliştirici her bir çevrimde riskleri anlamakta ve bu risklere göre önlem almaktadır. Böylece olası hataların önüne geçilmektedir (Sarıdoğan, 2017).

Hızlı uygulama geliştirme modelinin temel hedefi, yüksek kaliteli sistemleri, düşük maliyet ile hızlı geliştirme ve teslim etmekten oluşmaktadır (Beynon, 1999: 211). Gereksinimler iyi anlaşılmışsa ve proje kapsamı sınırlandırılmışsa, bu model ile kısa zaman dilimlerinde (örneğin 60 ile 90 gün) tam işlevli bir yazılım geliştirilebilmektedir.

Öngörülebilir bir program ve bütçe içerisinde son kullanıcıların ihtiyaçlarını karşılayan yüksek kaliteli yazılım geliştirme modeli olan rasyonel birleştirilmiş süreç modeli kullanmak isteyen kuruluşların gereksinimlerine uyacak şekilde uyarlanabilmekte ve genişletilebilmektedir. Başlangıç, olgunlaşma veya düzenleme, yapım ve geçiş olarak dört adımdan oluşmaktadır. Başlangıç aşamasında gereksinimler belirlenirken, olgunlaşma aşamasında riskler belirlenmekte ve bir prototip oluşturulmaktadır. (Kruchten, 2004).

Gereksinimlerin net olarak bilinmediği yazılım projelerinde kullanılan prototip modelde, ilk aşamada belirlenebilen gereksinimler belirlenir ve bir prototip oluşturulur. Oluşturulan prototip değerlendirildikten sonra yapılması gereken diğer gereksinimler belirlenir ve başka bir prototip daha oluşturulur. Gereksinimler tam olarak belirlenene kadar yazılımın yeni bir prototipi oluşturulur.

Gereksinimler tam olarak belli olduktan sonra, istenilen yazılım bütün olarak gerçekleştirilir (Gürbüz, 2010). Örneğin, yeni bir araba geliştirilirken, bir veya daha fazla prototip ayrı ayrı oluşturulacaktır. Bu prototipler, bir üretim hattı kurulmadan önce yoğun bir şekilde test edilir. Yazılım geliştirme ile benzer bir yaklaşım izlemek mümkündür. Prototipleme, bir sistemin ilk sürümünü inşa etme pratiğidir, bu da nihai sistemin tüm özelliklerini yansıtmaz (Bell, 2005).

Literatüre bakıldığında yazılım süreç modelleri konusunda birçok çalışmanın yapıldığı görülmektedir. Akyol 2008 yılında gerçekleştirdiği anket ile 2001 yılında Türkiye Kalite Derneği (KalDer) Yazılımda Toplam Kalite Yönetimi Uzmanlık Grubu tarafından gerçekleştirilen "Yazılım Sektörü Anketini" tekrar ederek, Türkiye'de yazılım mühendisliği ve yazılım yönetimi uygulamalarının o zamana göre son yedi yıl içerisindeki gelişimini ve mevcut olgunluk düzeyini değerlendirmiştir. Yazılım mühendisliği metotlarının kullanımının 2001-2008 yılları arasında yaklaşık %10'luk bir artış gösterdiği bu çalışma neticesinde bulunmuştur (Akyol, 2009).

Garousi ve arkadaşları Türkiye'deki yazılım mühendisliği uygulamaları üzerine bir çalışma yapmıştırlar. Çalışma neticesinde Türkiye'de yazılım geliştirme esnasında eski bir yöntem olmasına rağmen şelale yönteminin kullanımının yaygın olduğu bulunmuştur (Garousi, Coşkunçay, Can ve Demirörs, 2015: 148).

Gül ise işletme özellikleri, yazılım geliştirme sürecinde kullanılan modellerin genel durumu, proje yönetim süreci, geliştirme süreci ve yazılım geliştirme projelerinin başarı kriterlerinin olduğu beş bölümlük bir anket çalışması sonucunda yazılım geliştiricilerin yazılım mühendisliği metotları hakkında daha fazla akademik bilgi sahibi olmaları gerektiği sonucuna ulaşmıştır (Gül, 2006: 151).

Yazılım süreçlerinin karşılaştırılması ve seçimi amacıyla Hindistan yazılım sektöründe yapılan bir çalışmada (Mahanti, Neogi ve Bhattacherjee, 2012) yazılım süreç modellerinin; ihtiyaçların anlaşılma seviyesi, karmaşıklık derecesi, müşteri katılım seviyesi, beklenen risk derecesi, proje süresi, geliştiricilerin yeteneği ve proje büyüklüğü gibi kriterler dikkate alınmıştır. Çalışma sonucunda yazılım süreç modellerinin seçimi büyük ve karmaşık projeler, küçük ve orta projeler, her seviye projeler olmak üzere üç başlıkta sınıflandırılmıştır. Fakat çalışmada belirlenen kriterler içerisinde bulunan yazılım geliştiricilerin yeteneği kriteri içerisinde geliştiricilerin kullanacakları yazılım süreç modelleri hakkındaki farkındalık seviyelerine değinilmemiştir.

Yazılım süreç modellerinin seçimi ile ilgili yapılan bazı çalışmalarda ise maliyet, ülkeye özgü düzenlemeler, geçmiş tecrübe ve eğitim seviyesi ile müşteri ve yazılım şirketi tercihlerinin model seçimi üzerinde etkili olabileceği gösterilmiştir. Bir çalışmada da otuz beş parçalık kriter seti yazılımlara uygulanmış ve kullanılan yazılım süreç modeli bulunmuştur. Ayrıca herhangi bir proje için yeterli olan koşulların başka bir proje için yetersiz olabileceği sonucuna varılmıştır (Ocaktan ve Yıldıztekin, 2011). Yazılım yöneticileri ve geliştiricilerinin bir yazılım projesine başlamadan önce doğru model seçimi konusunda önerilere ihtiyaçları vardır. Bu yüzden yazılım süreç modelleri, kullanım yerleri, nedenleri ve oranları ile farkındalık seviyeleri üzerinde daha fazla çalışma yapılmasına ihtiyaç vardır.

Verilen literatüre göre yazılım geliştiricilerin yazılım mühendisliği yöntem bilimleri hakkında eksiklikleri bulunmaktadır. Bu çalışma ile yazılım geliştiricilerin yazılım mühendisliği yöntem bilimlerinin önemli konularından biri olan yazılım süreç modelleri hakkındaki bilgi düzeyleri ölçülmek istenmiş, yazılım geliştiricilerin yazılım süreç modelleri hakkındaki bilgi düzeylerinin nasıl artırılabileceğine dair öneriler sunulmuştur. Çalışmanın araştırma sorusu ise "Yazılım geliştiricilerin yazılım süreç modellerini kullanım durumları ile yazılım süreç modelleri farkındalık yanıtları arasında anlamlı bir farklılık var mıdır?" şeklindendir.

2. YÖNTEM

Yazılım geliştiricilerinin bir ürünü ortaya koymak için yazılım mühendisliği yöntemlerini ne derecede kullandıklarını ve kullandıkları yöntemlerin farkında olma düzeylerini ortaya koyabilmek için "Yazılım Geliştiricilerin Yazılım Süreç Modellerini Kullanma Farkındalıkları Düzeyini Belirleme Anketi" oluşturulmuştur (Ek 1). Anket, kullanılan yazılım geliştirme modellerinin aşamaları, birbirine göre farkları ve üstünlükleri başta olmak üzere, literatürdeki çalışmalar incelenerek ve uzmanlardan görüş alınarak oluşturulmuştur.

Oluşturulan anket demografik bilgiler bölümü ve yazılım süreç modelleri üzerine yöneltilmiş sorular bölümü olmak üzere iki bölümden oluşmaktadır. Çalışma kapsamında; kod eksenli yazılım geliştirme, şelale modeli, V modeli, artımlı geliştirme modeli, evrimsel model, spiral model, hızlı uygulama geliştirme modeli, rasyonel birleştirilmiş süreç modeli, prototip model olmak üzere dokuz ayrı yazılım geliştirme modeli üzerinde durulmuştur. Bu anket; modellerin aşamaları, farkları ve birbirleri üzerindeki üstünlükleri göz önünde tutulmuş ve beşli likert ölçeği şeklinde 23 adet sorudan oluşmaktadır. Ölçek içerisinde yer alan modeller, soru sayıları ve Ek 1'de yer alan soru numaraları Tablo 1'de gösterilmiştir.

Bu aşamadan sonra elde edilen ölçek, seçkisiz olmayan örnekleme yaklaşımlarından amaçsal örnekleme (Büyüköztürk, Çakmak, Akgün, Karadeniz ve Demirel, 2014) ile oluşturulmuş toplamda kırk dört katılımcıya uygulanmıştır. Katılımcılar Atatürk Üniversitesi'nin farklı fakülte ve merkezlerindeki yazılım geliştirme biriminde görev yapan akademik ve teknik kadroda görev yapan personellerden oluşmaktadır. Ölçeği uygulama aşamasında örneklem grubuna anket hakkında ek bir açıklama yapılmamış olup, yazılım geliştirme modelleri terimlerini bildikleri varsayılmıştır. Değerlendirme aşamasında ise örneklem grubundan toplanan veriler analiz edilmiştir.

Yazılım Geliştirme Modeli	Soru Sayısı	Soru Numarası (Ek 1)
V Modeli	2	1-2
Şelale Modeli	5	3-7
Prototip Model	2	8-9
Spiral Model	2	10-11
Evrimsel Model	3	12-14
Artımlı Geliştirme Modeli	3	15-17
Kod Eksenli Geliştirme Modeli	1	18
Hızlı Uygulama Geliştirme Modeli	1	19
Rasyonel Birleştirilmiş Süreç Modeli	4	20-23

Tablo 1. Yazılım Geliştiricilerin Yazılım süreç modellerini Kullanma Farkındalıkları Düzeyini Belirleme Anketi İçerisinde Bulunan Model Bilgileri

Örneklem grubundan elde edilen veriler bağımsız örneklem t testi ile analiz edilmiştir. Bağımsız örneklem t testi her bir model için ayrı ayrı uygulanmış olup katılımcıların hangi model üzerinde kullanım farkındalıklarının olup olmadığı tespit edilmiştir.

3. BULGULAR

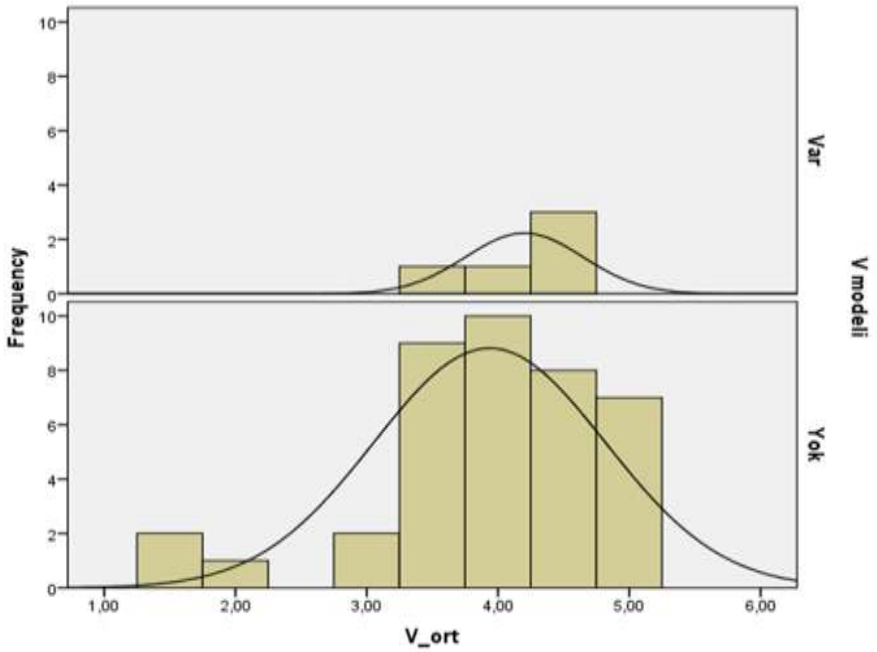
V modelini kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin V modeline yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 2.'de sunulmuştur.

Değişken	Grup	N	$\bar{X}$	SS	t	sd	p
V Modeli	V modelini kullanan	5	4.2	.45	.65	4	.51
	V modelini kullanmayan	39	3.93	.88	1.07	38	.31

Tablo 2. V Modelini Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin, V Modeli Yargılarına Katılma Düzeyi Ortalamaları Arasındaki Farklılık

Tablo 2. incelendiğinde bağımsız örneklem t testi sonuçlarına göre V modelini kullandığını ifade eden yazılım geliştiricilerin V modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$ = 4.2) ile V modelini kullanmadığını ifade eden yazılım geliştiricilerin V modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.93) arasında istatistiksel olarak anlamlı bir farklılık görülmemiştir [t(4)=.65, p>.05], [t(38)=1.07, p>.05].

V modelini kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin V modeline yönelik yargılara katılma düzeyi ortalamaları ve dağılımları sonucunda elde edilen veriler Grafik 1.'de sunulmuştur.



Grafik 1. V Modelini Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

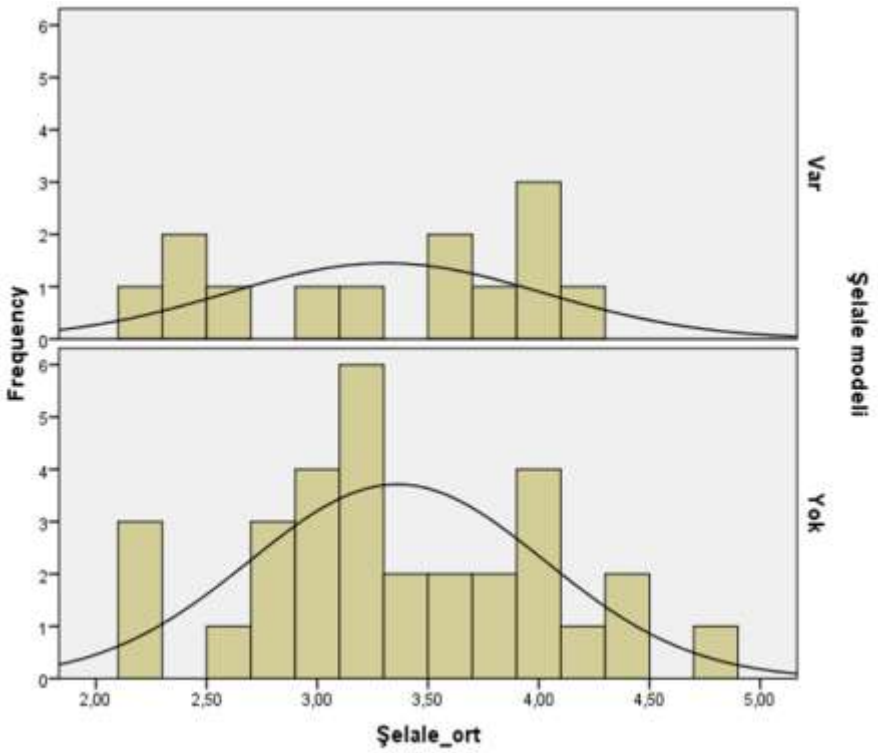
Şelale modelini kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin şelale modeline yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 3.'de sunulmuştur.

Değişken	Grup	N	$\bar{X}$	SS	t	sd	p
Şelale Modeli	Şelale modelini kullanan	13	3.31	.61	-.23	42	.81
	Şelale modelini kullanmayan	31	3.36	.67			

Tablo 3. Şelale Modelini Kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin, şelale modeli yargılarına katılma düzeyi ortalamaları arasındaki farklılık

Tablo 3. incelendiğinde bağımsız örneklem t testi sonuçlarına göre şelale modelini kullandığını ifade eden yazılım geliştiricilerin şelale modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$ = 3.31) ile şelale modelini kullanmadığını ifade eden yazılım geliştiricilerin şelale modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.36) arasında istatistiksel olarak anlamlı bir farklılık görülmemiştir [t(42)=-.23, p>.05].

Şelale modelini kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin şelale modeline yönelik yargılara katılma düzeyi ortalama verileri Grafik 2.'de sunulmuştur.



Grafik 2. Şelale Modelini Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

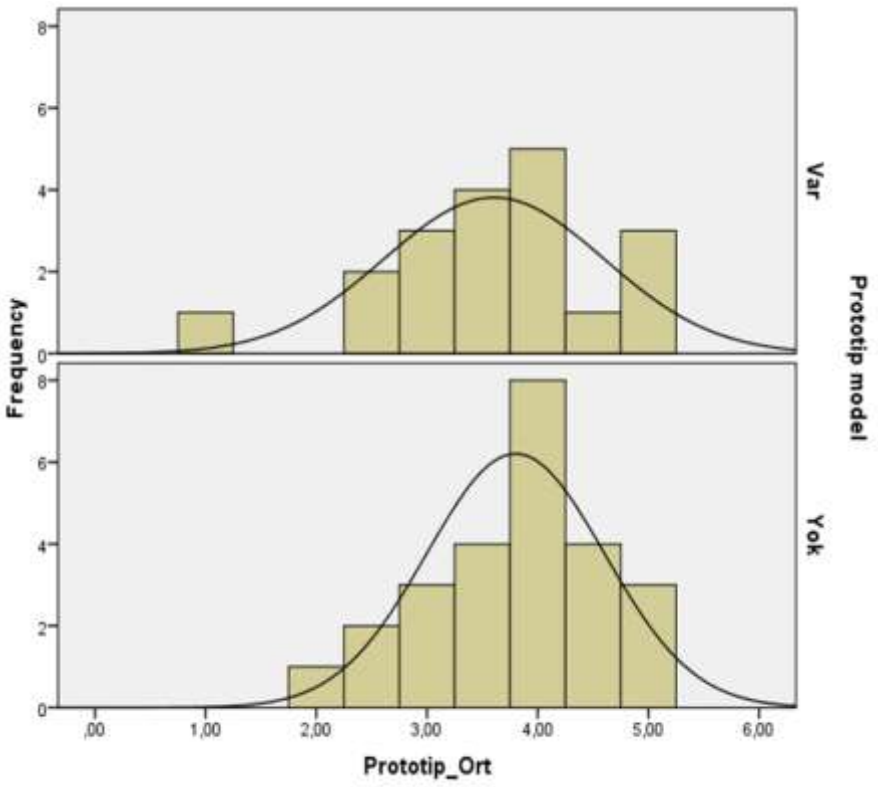
Prototip modeli kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin Prototip modele yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 4.de sunulmuştur.

Değişken	Grup	N	X̄	SS	t	sd	p
Prototip Model	Prototip modeli kullanan	19	3.60	.99	-.70	42	.48
	Prototip modeli kullanmayan	25	3.80	.80			

Tablo 4. Prototip Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin, Prototip Modeli Yargılarına Katılma Düzeyi Ortalamaları Arasındaki Farklılık

Tablo 4. incelendiğinde bağımsız örneklem t testi sonuçlarına göre Prototip modeli kullandığını ifade eden yazılım geliştiricilerin Prototip model yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.60) ile Prototip modeli kullanmadığını ifade eden yazılım geliştiricilerin Prototip model yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.80) arasında istatistiksel olarak anlamlı bir farklılık görülmemiştir [t(42)=-.70, p>.05].

Prototip modeli kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin Prototip modele yönelik yargılara katılma düzeyi ortalamalarının sonucunda elde edilen veriler Grafik 3.'de sunulmuştur.

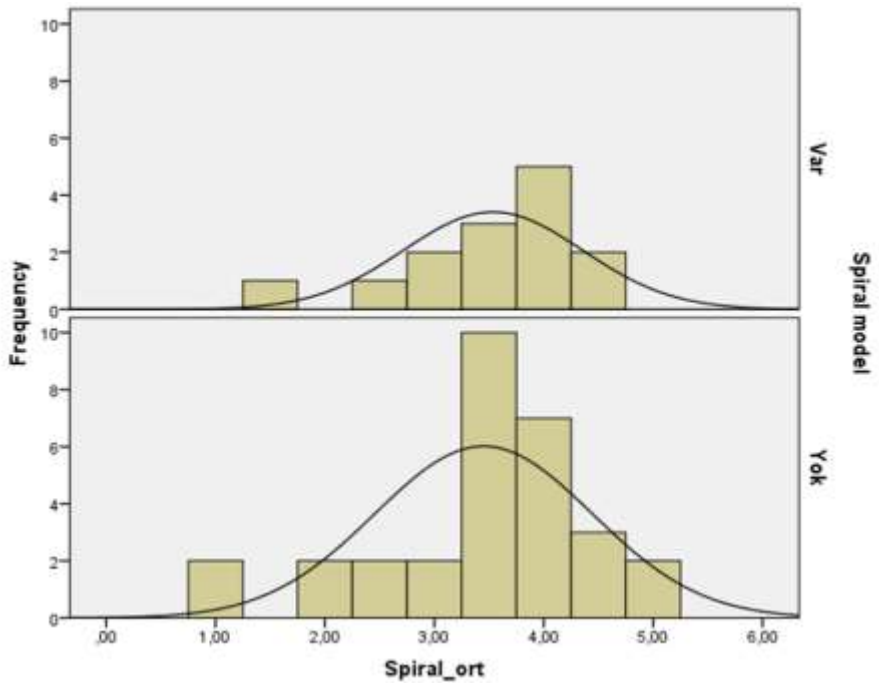


Grafik 3. Prototip Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

Spiral modeli kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin Spiral modele yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 5. ve Grafik 4.'de sunulmuştur.

Değişken	Grup	N	$\bar{X}$	SS	t	sd	p
Spiral Modeli	Spiral modeli kullanan	14	3.53	.81	.29	42	.77
	Spiral modeli kullanmayan	30	3.45	.99			

Tablo 5. Spiral Modeli Kullandığını Ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin, Spiral Model Yargılarına Katılma Düzeyi Ortalamaları Arasındaki Farklılık



Grafik 4. Spiral Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

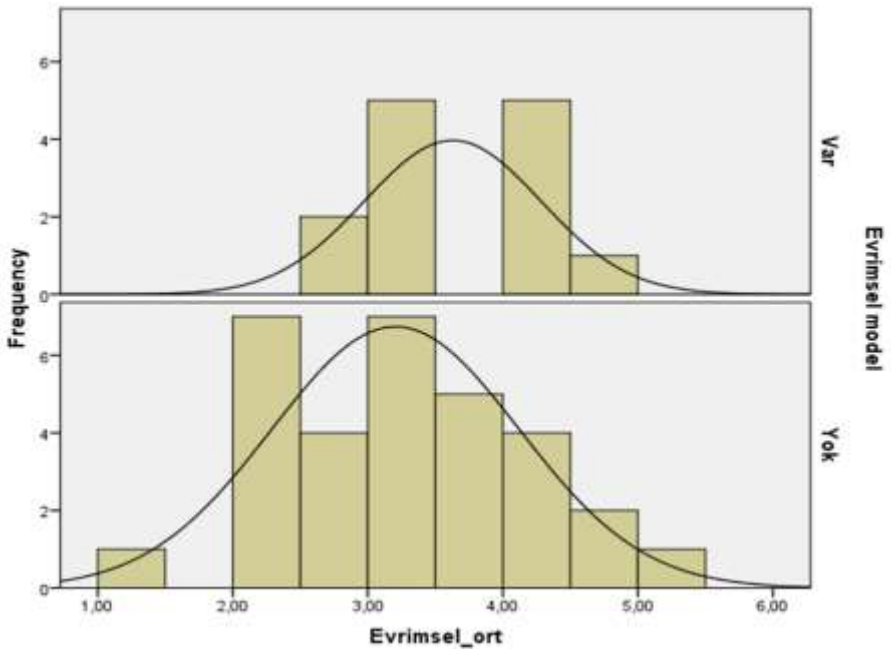
Tablo 5. incelendiğinde bağımsız örneklem t testi sonuçlarına göre Spiral modeli kullandığını ifade eden yazılım geliştiricilerin Spiral modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.53) ile Spiral modelini kullanmadığını ifade eden yazılım geliştiricilerin Spiral modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.45) arasında istatistiksel olarak anlamlı bir farklılık görülmemiştir [t(42)=.29, p>.05].

Evrimsel modeli kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin Evrimsel modele yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 6. ve Grafik 5.'de sunulmuştur.

Değişken	Grup	N	$\bar{X}$	SS	t	sd	p
Evrimsel Model	Evrimsel modeli kullanan	13	3.62	.65	1.50	12	.14
	Evrimsel modeli kullanmayan	31	3.20	.91	1.73	30	.091

Tablo 6. Evrimsel Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin, Evrimsel Modeli Yargılarına Katılma Düzeyi Ortalamaları Arasındaki Farklılık

Tablo 6. incelendiğinde bağımsız örneklem t testi sonuçlarına göre Evrimsel modeli kullandığını ifade eden yazılım geliştiricilerin Evrimsel model yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.62) ile Evrimsel modelin kullanmadığını ifade eden yazılım geliştiricilerin Evrimsel model yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.20) arasında istatistiksel olarak anlamlı bir farklılık görülmemiştir [t(12)=1.50, p>.05], [t(30)=1.73, p>.05].



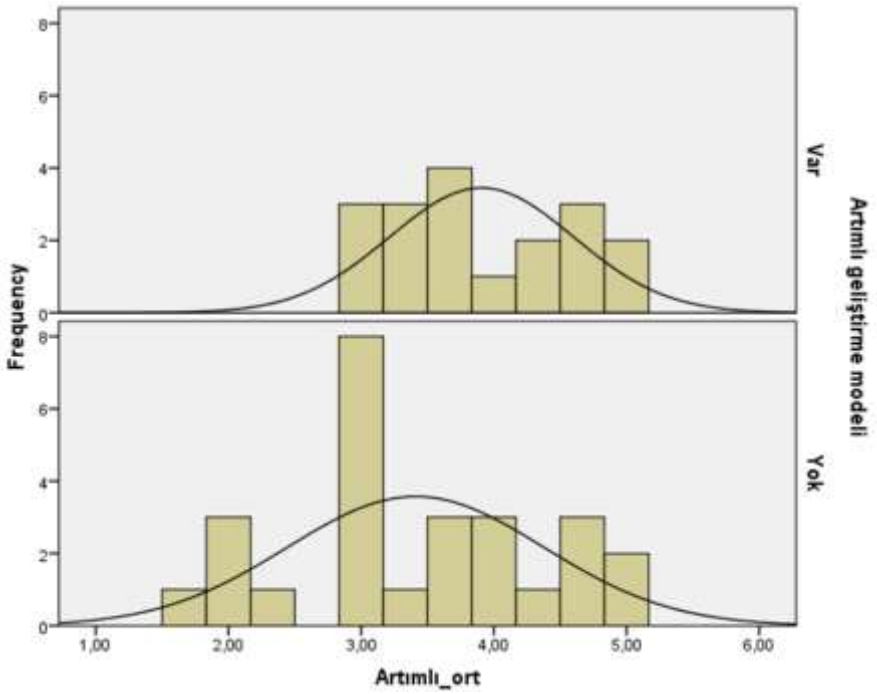
Grafik 5. Evrimsel Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

Artımlı geliştirme modelini kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin Artımlı geliştirme modeline yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 7. ve Grafik 6.'da sunulmuştur.

Değişken	**Grup**	**N**	**X̄**	**SS**	**t**	**sd**	**p**
Artımlı Geliştirme Modeli	Artımlı geliştirme modelini kullanan	18	3.90	.69	1.87	17	.06
	Artımlı geliştirme modelini kullanmayan	26	3.41	.96	1.98	25	

Tablo 7. Artımlı Geliştirme Modelini Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin, Artımlı Geliştirme Modeli Yargılarına Katılma Düzeyi Ortalamaları Arasındaki Farklılık

Tablo 7. incelendiğinde bağımsız örneklem t testi sonuçlarına göre Artımlı Geliştirme modelini kullandığını ifade eden yazılım geliştiricilerin Artımlı Geliştirme modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.90) ile Artımlı Geliştirme modelini kullanmadığını ifade eden yazılım geliştiricilerin Artımlı Geliştirme modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.41) arasında istatistiksel olarak anlamlı bir farklılık görülmemiştir [t(17)=1.87, p>.05], [t(25)=1.98, p>.05].



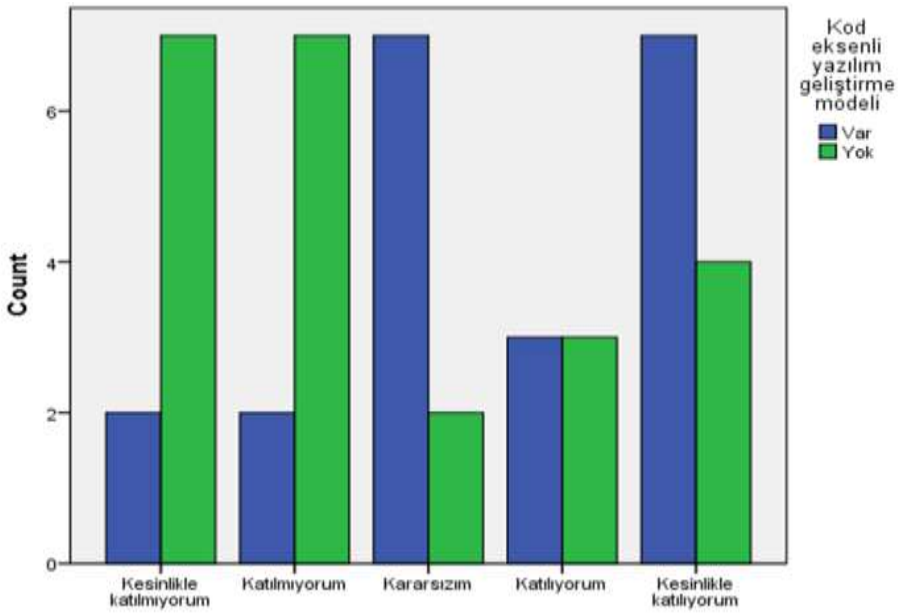
Grafik 6. Artımlı Geliştirme Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

Kod eksenli yazılım geliştirme modelini kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin Kod eksenli yazılım geliştirme modelini modeline yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 8. ve Grafik 7.'de sunulmuştur.

Değişken	Grup	N	$\bar{X}$	SS	t	sd	p
Kod Eksenli Yazılım Geliştirme Modeli	Kod eksenli yazılım geliştirme modelini kullanan	21	3.52	1.32	2.22	42	.03
	Kod eksenli yazılım geliştirme modelini kullanmayan	23	2.56	1.50			

Tablo 8. Kod Eksenli Yazılım Geliştirme Modelini Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin, Kod Eksenli Yazılım Geliştirme Modeli Yargılarına Katılma Düzeyi Ortalamaları Arasındaki Farklılık

Tablo 8. incelendiğinde bağımsız örneklem t testi sonuçlarına göre Kod eksenli yazılım geliştirme modelini kullandığını ifade eden yazılım geliştiricilerin Kod eksenli yazılım geliştirme modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$ = 3.52) ile Kod eksenli yazılım geliştirme modelini kullanmadığını ifade eden yazılım geliştiricilerin Kod eksenli yazılım geliştirme modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=2.56) arasında istatistiksel olarak anlamlı ve Kod eksenli yazılım geliştirme modelini kullananlar lehine bir farklılık görülmüştür [t(42)=2.22, p<.05].



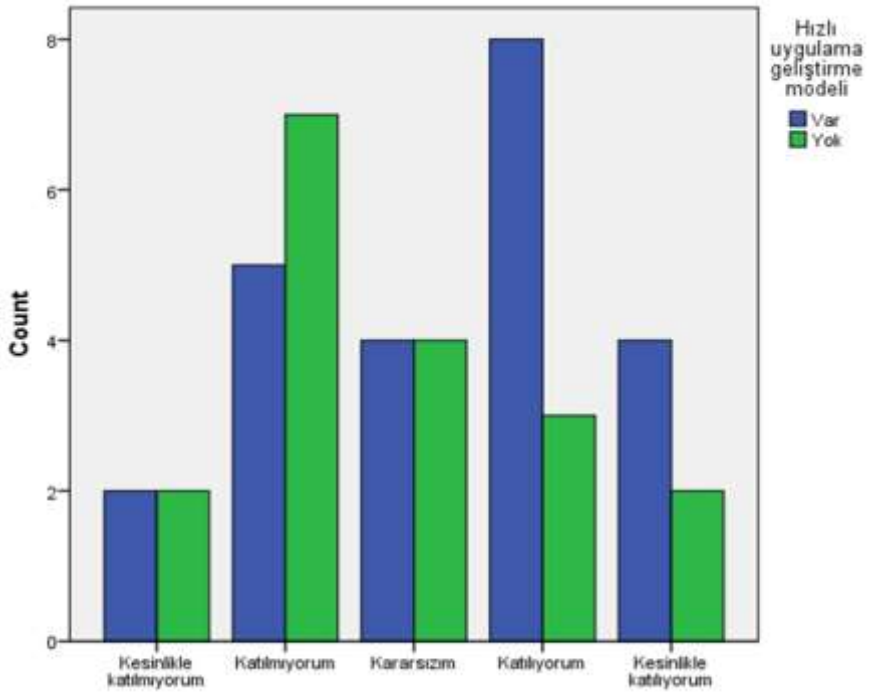
Grafik 7. Kod Eksenli Yazılım Geliştirme Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

Hızlı uygulama geliştirme modelini kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin Hızlı uygulama geliştirme modeline yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 9. ve Grafik 8.'de sunulmuştur.

Değişken	Grup	N	$\bar{X}$	SS	t	sd	p
Hızlı Uygulama Geliştirme Modeli (RAD)	RAD modelini kullanan	23	3.30	1.25	1.35	39	.18
	RAD modelini kullanmayan	18	2.77	1.21			

Tablo 9. Hızlı Uygulama Geliştirme Modelini Kullandığını Ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin, Hızlı Uygulama Geliştirme Modeli Yargılarına Katılma Düzeyi Ortalamaları Arasındaki Farklılık

Tablo 9. incelendiğinde bağımsız örneklem t testi sonuçlarına göre RAD modelini kullandığını ifade eden yazılım geliştiricilerin RAD modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.30) ile RAD modelini kullanmadığını ifade eden yazılım geliştiricilerin RAD yargılarına katılma düzeyi ortalaması ($\bar{X}$=2.77) arasında istatistiksel olarak anlamlı bir farklılık görülmemiştir [t(39)=1.35, p>.05].



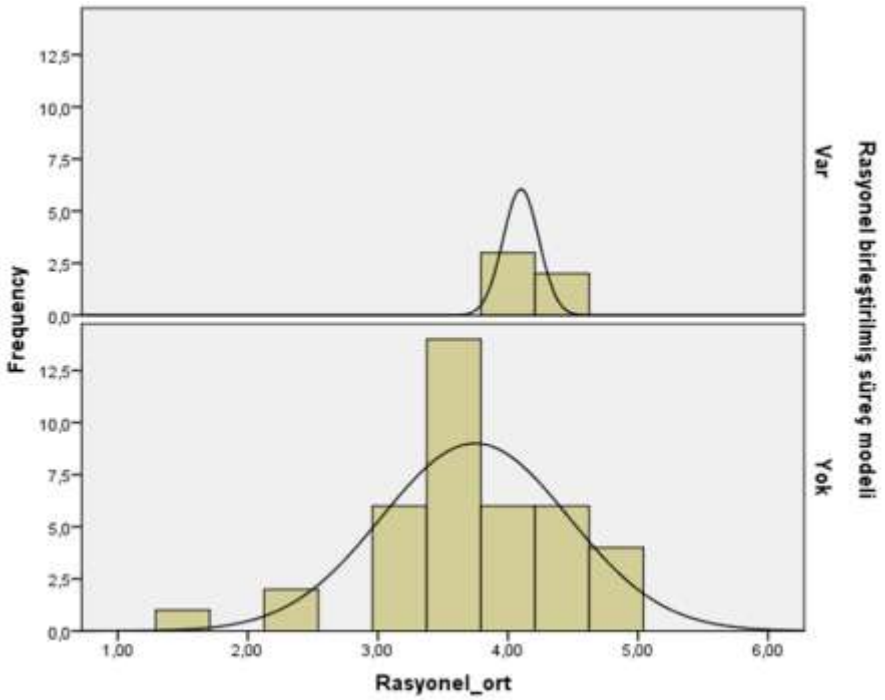
Grafik 8. RAD Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

Rasyonel Birleştirilmiş Süreç (RUP) modelini kullandığını ve kullanmadığını ifade eden yazılım geliştiricilerin Rasyonel Birleştirilmiş Süreç modeline yönelik yargılara katılma düzeyi ortalamalarının farklı olup olmadığını tespit etmek için yapılan bağımsız örneklem t testi analizi sonucunda elde edilen veriler Tablo 10. ve Grafik 9.'da sunulmuştur.

Değişken	Grup	N	$\bar{X}$	SS	t	sd	p
RUP Modeli	RUP modelini kullanan	5	4.1	.14	1.08	4	.29
	RUP modelini kullanmayan	39	3.74	.72	2.70	38	.10

Tablo 10. Rasyonel Birleştirilmiş Süreç Modelini Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin, Rasyonel Birleştirilmiş Süreç Modeli Yargılarına Katılma Düzeyi Ortalamaları Arasındaki Farklılık

Tablo 10. incelendiğinde bağımsız örneklem t testi sonuçlarına göre RUP modelini kullandığını ifade eden yazılım geliştiricilerin RUP modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=4.1) ile RUP modelini kullanmadığını ifade eden yazılım geliştiricilerin RUP modeli yargılarına katılma düzeyi ortalaması ($\bar{X}$=3.74) arasında istatistiksel olarak anlamlı bir farklılık görülmemiştir [t(4)=1.08, p>.05], [t(38)=2.70, p>.05].



Grafik 9. RUP Modeli Kullandığını ve Kullanmadığını İfade Eden Yazılım Geliştiricilerin Model Yargılarına Katılma Düzeyleri

4. SONUÇ VE ÖNERİLER

Çalışmanın amacı yazılım geliştiricilerin bir yazılımı geliştirirken hangi yazılım süreç modeli kullandıklarının farkında olup olmadıklarını belirlemek ve bu farkındalık durumuna göre yazılım geliştirme biriminde görev yapan bireylere önerilerde bulunmaktır.

Atatürk Üniversitesinin farklı akademik ve teknik birimlerinde, yazılım geliştirme işinin yapıldığı yerlerde (BAUM, AÖF, UZEM, Bilgisayar Müh. Bölümü, Yönetim Bilişim Sistemleri Bölümü) görev yapan 44 katılımcıdan anket yöntemi ile elde edilen veriler bağımsız örneklem t testi uygulanarak analiz edilmiştir. Katılımcılar ankette yer alan V modeli, şelale modeli, prototip model, spiral model, evrimsel model, artımlı yazılım geliştirme modeli, kod eksenli yazılım geliştirme modeli, hızlı uygulama geliştirme modeli ve rasyonel birleştirilmiş süreç modellerinin adımları ve uygulanma biçimlerine yönelik oluşturulmuş yargılara katılma düzeylerini ifade etmişlerdir.

Katılımcıların ankette bulunan yazılım geliştirme model adımlarına belirttikleri katılma düzeylerinin analizlerine göre; ankette yer alan kod eksenli yazılım geliştirme modeli adımlarına yönelik yargılara katılımcıların katılma düzeyleri ifade sonuçları arasında anlamlı bir fark olduğu gözlemlenmiştir. Yani örneklem grubunda yer alan katılımcılar bir projede yazılım geliştirirken kod eksenli yazılım geliştirme modeli adımlarını kullandıklarını ve bunun da farkında oldukları sonucuna ulaşılmıştır.

Ankette yer alan V modeli, şelale modeli, prototip model, spiral model, evrimsel model, artımlı yazılım geliştirme modeli, hızlı uygulama geliştirme modeli ve rasyonel birleştirilmiş süreç modeli yazılım geliştirme adımlarına yönelik yargılara katılımcıların katılma düzeyleri ifade sonuçları arasında anlamlı bir fark olmadığı gözlemlenmiştir. Yani örneklem grubunda yer alan katılımcılar bir projede yazılım geliştirirken adı geçen yazılım geliştirme modellerinin adımlarını kullandıklarını fakat bunun farkında olmadıkları görülmüştür. Diğer bir ifadeyle katılımcıların bu modellere yönelik katılım düzeyleri ile bu modelleri kullanma durumları arasında negatif bir durum söz konusudur.

Genel olarak kod eksenli yazılım geliştirme modeli dışındaki modellerin adımlarına katılma düzeyleri yüksek olan adaylar bu modelleri kullandıklarının farkında olmadıkları veya adımlara katılma düzeyleri düşük olan katılımcıların da bu modelleri kullandıkları yanılgısına düştükleri sonucuna ulaşılmıştır.

Araştırma bulguları dikkate alındığında; başarılı yazılım projeleri gerçekleştirme bağlamında yazılım geliştiricilere yazılım kalitesi ve yazılım süreç modelleri konusunda şunlar önerilebilir.

Yazılım projelerini yöneten kişiler eğer yazılım alanı dışından iseler temel proje yönetim metotları yanında yazılım mühendisliğinin temel paradigmaları konusunda da bilgi edinmeleri gerekir.

Yazılım projeleri, süreç iyileştirmeye dayalı kalite anlayışı ile tanımlanıp geliştirilmelidir.

Yazılım geliştiriciler, var olan yazılım geliştirme bilgileri yanında yazılım mühendisliğinin temel konularından olan yazılım süreçleri ve yazılım süreç modelleri hakkında eğitimler almalıdırlar.

Yazılım süreç modelleri hakkında farkındalık oluşturacak bilgiler tekrarlanmalıdır.

EK 1. Anket Formu

YAZILIM GELİŞTİRİCİLERİN YAZILIM SÜREÇ MODELLERİNİ KULLANMA FARKINDALIKLARI DÜZEYİNİ BELİRLEME ANKETİ

Değerli katılımcı,

Bu anket, Atatürk Üniversitesi, Sosyal Bilimleri Enstitüsü, Yönetim Bilişim Sitemleri Bölümü'nde yürütülen "Yazılım Geliştiricilerin Yazılım süreç modelleri Kullanma Farkındalıkları" başlıklı çalışma kapsamında Atatürk Üniversitesinde bulunan yazılım geliştirme biriminde görev yapan personellerin yazılım geliştirirken kullandıkları yazılım süreç modellerinin farkındalıklarını ölçmek amacıyla oluşturulmuştur. Anket demografik bilgiler ve yazılım süreç modelleri farkındalığını oluşturan sorular olmak üzere iki bölümden oluşmaktadır. Kişisel bilgiler 3. Şahıslarla paylaşılmayacak olup toplanan bütün veriler bilimsel amaçlı kullanılacaktır. Ankete katılım gönüllülük esasına dayanmaktadır. Zaman ayırdığınız için teşekkür ederiz.

DEMOGRAFİK BİLGİLER

1. Cinsiyetiniz?

Kadın ( ) Erkek ( )

2. Yaşınız?

19 – 24 ( ) 25 – 30 ( ) 29 – 34 ( ) 35 – 40 ( ) 40+ ( )

3. Eğitim Durumunuz?

Lise ( ) Önlisans ( ) Lisans ( ) Yüksek Lisans ( ) Doktora ( )

4. Atatürk Üniversitesinde hangi birimde çalışıyorsunuz?

…….

5. Çalıştığınız birimde kaç yıldır görev yapmaktasınız? ……

6. Kaç yıldır yazılım geliştirmeyle ilgileniyorsunuz? ……

Yazılım Süreç Modelleri Farkındalığına Yönelik Sorular

Bu bölümdeki maddelerde belirtilen kriterleri yazılım geliştirme modellerinde bulunan özelliklere yönelik sorular bulunmaktadır. Maddelere katılma düzeyiniz şu şekildedir; (1) Kesinlikle katılmıyorum (2) Katılmıyorum (3) Kararsızım (4) Katılıyorum (5) Kesinlikle katılıyorum

1	Testler sırasında bulunan hataların düzeltilmesi için hangi düzeye dönülmesi gerektiğini biliyorum.	(1)	(2)	(3)	(4)	(5)
2	Yazılım geliştirirken isterleri iyi tanımlarım, az belirsizlikle aşamalar halinde ilerlerim.	(1)	(2)	(3)	(4)	(5)
3	Yazılımı teslim ederken hangi isterleri yerine getireceğini belgelendirmeden tasarıma başlamam.	(1)	(2)	(3)	(4)	(5)
4	Yazılım geliştirme sürecini; analiz, tasarım, geliştirme ve test aşamalarına ayırır ve bunları ardışık olarak gerçekleştiririm.	(1)	(2)	(3)	(4)	(5)
5	Yazılım geliştirmemi isteyen kurum veya kişilerle yazılımın ne yapacağı konusunu en başta ve sadece bir kere konuşurum.	(1)	(2)	(3)	(4)	(5)
6	Yazılımın ne yapacağını belirledikten sonra kullanıcılara ilerleme ile ilgili bilgi vermem.	(1)	(2)	(3)	(4)	(5)
7	Çözümleme ve tasarım süreçlerinden ziyade kendimi kod yazmaya daha eğilimli hissediyorum.	(1)	(2)	(3)	(4)	(5)
8	Yazılımı isteyen kullanıcı ile sistemin genel isterlerini tanımlayarak ve odaklanılması gereken noktaları belirlerim.	(1)	(2)	(3)	(4)	(5)
9	Geliştireceğim yazılım tamamlanmadan sadece bir ön ürünü kullanıcıya sunarım.	(1)	(2)	(3)	(4)	(5)
10	Yazılım geliştirirken riskleri tanımlayarak olası çözüm yöntemlerini irdelerim.	(1)	(2)	(3)	(4)	(5)
11	Yazılım geliştirirken kullanıcı her aşamadan haberdardır.	(1)	(2)	(3)	(4)	(5)
12	Yazılım geliştirirken her aşamada bir ön ürün ortaya çıkarırım.	(1)	(2)	(3)	(4)	(5)
13	Yazılım geliştirme aşamalarının her birinde kullanıcıya bir deneme sürümü sunarım.	(1)	(2)	(3)	(4)	(5)
14	Kullanıcı istediği yazılımı tanımlayamıyorsa, kullanıcının temel isteklerine göre bir çözüm geliştirerek gerçek isteklerini ortaya çıkarırım.	(1)	(2)	(3)	(4)	(5)

15	Yazılım geliştirirken ilk olarak temel işlevleri gerçekleştiren bir ön ürün ortaya çıkarırım.	(1)	(2)	(3)	(4)	(5)
16	Yazılımı ayrı ayrı çalışabilecek parçalara ayırarak geliştiririm.	(1)	(2)	(3)	(4)	(5)
17	Son ürünü oluşturan her bir parçayı, nihai ürünün daha güvenilir ve işlevsel olması için test ederim.	(1)	(2)	(3)	(4)	(5)
18	Yazılım geliştirmeye başlarken ilk olarak kodlama işlemi yaparım.	(1)	(2)	(3)	(4)	(5)
19	Yazılım geliştirirken bir an önce çalışan bir uygulama oluşturmak isterim.	(1)	(2)	(3)	(4)	(5)
20	Yazılım geliştirirken ihtiyaç duyulan dokümanlar için önceden oluşturulmuş hazır şablonları kullanırım.	(1)	(2)	(3)	(4)	(5)
21	Yazılım geliştirirken daha önceki kütüphanelerden faydalanırım.	(1)	(2)	(3)	(4)	(5)
22	Yazılım geliştirirken çalışanların görev ve sorumlulukları net bir şekilde tanımlarım.	(1)	(2)	(3)	(4)	(5)
23	Yazılım geliştirirken süreç yönetim araçlarını kullanırım.	(1)	(2)	(3)	(4)	(5)

8. Yazılım geliştirirken kullandığınız model/leri seçiniz. (Bir ya da daha fazla seçenek işaretleyebilirsiniz)

Kod Eksenli Yazılım Geliştirme Modeli ( )

Şelale Modeli ( )

Artımlı Geliştirme Modeli ( )

Rasyonel Birleştirilmiş Süreç Modeli ( )

Spiral Model ( )

Evrimsel Model ( )

Hızlı Uygulama Geliştirme Modeli ( )

V Modeli ( )

Prototip Model ( )

KAYNAKÇA

Acuna, S. T., De Antonio, A., Ferre, X., Lopez, M., Mate, L., & Estero, S. (2000). The Software Process: Modelling, Evaluation and Improvement. Handbook of Software Engineering and Knowledge Engineering, Vol. 0, No. 0 (2000) 000-000, World Scientific Publishing Company.

Akyol, M.M. (2009). Türkiye'de Yazılım Mühendisliği ve Yazılım Yönetimi Uygulamaları, Fen Bilimleri Enstitüsü Bilgi Teknolojileri Programı. Bahçeşehir Üniversitesi: İstanbul.

Aysolmaz, B., Yıldız, H., Demirörs, O. (2011). Ulusal Yazılım Mühendisliği Sempozyumu, 163-168.

Bell, D. (2005). Software Engineering for Students: a Programming Approach. Pearson Education.

Beynon-Davies, P. (1999). Rapid Application Development (RAD): An Empirical Review. European Journal of Information Systems. 1999. 8(3): 211-223.
Bourque, P., & Fairley, R. E. (2004). Guide to The Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.
Boehm, B.W. (1988). A spiral Model of Software Development and Enhancement. Computer. 21(5): 61-72.
Büyüköztürk, Ş., Çakmak, E., Akgün, Ö. E., Karardeniz, Ş. ve Demirel, F. (2014). Bilimsel Araştırma Yöntemleri. Ankara: Pegem Akademi.
Erten, M. (2016). Temel Kavramlar. R. Çölkesen (Dü.) içinde, Bilgisayar Mühendisliğine Giriş (s. 15-32). İstanbul: Papatya Yayıncılık .
Garousi, V., Coşkunçay, A., Can, A. B., Demirörs, O. (2015). A Survey of Software Engineering Practices in Turkey. Journal of Systems and Software. 108: 148-177.
Gül, Z. (2006). Yazılım Geliştirme Sürecinin İyileştirilmesi ve Türkiye Uygulamaları, Fen Bilimleri Enstitüsü İşletme Mühendisliği Anabilim Dalı. 2006, İstanbul Teknik Üniversitesi: İstanbul. 151.
Gürbüz, A. (2010). Yazılım Test Mühendisliği. İstanbul: Papatya Yayıncılık Eğitim. 176.
Kruchten, P. (2004). The Rational Unified Process: An Introduction. Addison-Wesley.
Mahanti R., Neogi M.S., Bhattacherjee V. (2012). Factors Affecting the Choice of Software Life Cycle Models in the Software Industry-An Empirical Study. Journal of Computer Science. Volume 8. s. 1253-1262.
Munassar N.M.A., Govardhan A. (2012). A Comparison Between Five Models Of Software Engineering. IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 5, 94-101
Nizam, A. (2015). Yazılım Proje Yönetimi. 2 ed. İstanbul: Papatya Yaayıncılık Eğitim. 490.
Ocaktan Ş. ve Yıldıztekin M. (2011). Güvenli Yazılım Geliştirme Süreç Modellerinin Karşılaştırılması Uygulamaları. Elektrik – Elektronik ve Bilgisayar Sempozyomu. Elazığ.
Pauca, V.P (2003). Software Life Cycle, Wake Forest University, CSC 331/631, Spring.
Pressman, R.S. (2010). Software Engineering: A Practitioner's Approach. 7 ed. Raghothaman Srinivasan. 895.
Radatz, J., A. Geraci, and F. Katki (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE Std, 1990. 610121990(121990): 84.
Rehman. A, Hussain. R (2007). Software Project Management Methodologies/Frameworks Dynamics "A Comparative Approoach". In Information and Emerging Technologies International Conference on (pp. 1-5). IEEE.
Sarıdoğan, M.E. (2017). Yazılım Mühendisliği : Yöntembilimleri, Standartları ve Belgeleriyle 3ed. İstanbul: Papatya Yayıcılık Eğitim A.Ş. 568.
Scacchi, W. (2001). Process Models in Software Engineering. Encyclopedia of Software Engineering. 2. Ed., John Wiley and Sons, Inc, New York.
Sharma, B., Sharma, N., & Sharma, N. (2009, December). Software Process Improvement: A Comparative Analysis of SPI models. In Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on (pp. 1019-1024). IEEE.
Sommervile, I. (2000). Software Engineering, New York: Addison-Wesley, Harlow, England.
Sommerville, I. (2011). Software engineering (9'th Edition). USA: Pearson Education,Inc.
Stephens, R. (2015). Beginning Software Engineering. John Wiley & Sons. ISBN: 978-1-118-96914-4.
The Standish Group International. (2009). Inc., CHAOS Report.