PTP-Vision: Bankacılık Uygulamaları için Proxy Tabanlı Kişisel Yazılım Test Eforu Tahmin Metodu

Gizem Kahveci[1] Kıvanç Dinçer[2][0000-0002-4051-4968]

[1] Hacettepe Üniversitesi, Ankara, TR
[2] İstanbul Gelişim Üniversitesi, İstanbul, TR
gizemkhv@gmail.com kdincer@gelisim.edu.tr

Özet. Bu çalışmada özel bir bankanın yazılım proje yöneticilerine yönelik olarak, kodlama ve birim testleri tamamlanan yazılım birimlerinin fonksiyonel testleri için gerekecek test süresi/eforunu tahmin ederlerken kullanabilecekleri özgün bir yöntem geliştirilmesi amaçlanmıştır. *PTP-Vision* adı verilen bu yöntem ile, bir test analistine atanan her bir yazılım birimi için gerekecek test çalıştırma kişisel eforunun (dolayısıyla süresini) test sürecinin ön safhalarında iken doğru bir şekilde tahmin edilebilmesi sağlanmıştır. Geliştirilen yöntemin temelinde, test analistinin kendi kişisel test sürecini Humphrey'in Kişisel Yazılım Süreci'nde (*Personal Software Process - PSP*) tanımlandığı gibi modelleyerek ölçümlemesi ve bu ölçümlerin analizi suretiyle kişisel tahmin veritabanını oluşturması bulunmaktadır. Bu yöntemin geliştirme ve sınama çalışmaları için bankanın yazılım uygulamalarından elde edilen gerçek veriler kullanılmıştır. Geliştirilen yöntemin çalışabilirliği ve hassasiyeti test sürecinin ön safhalarında yapılan tahminler ile gerçekleşen efor/süre değerleri karşılaştırılmak suretiyle değerlendirilmiştir. Ortam ve uygulama ile ilgili belli parametrelerin aynı kalması durumunda tahmin hatalarının %12 bandını aşmadığı, çoğunlukla çok daha küçük olduğu görülmüştür.

Anahtar Kelimeler: Test süresi tahmini, Test eforu tahmini, Proxy-tabanlı tahmin, Kişisel test süreci, Bankacılık yazılım uygulamaları.

PTP-Vision: A Proxy-Based Personal Software Test Effort Estimation Method for Banking Applications

Gizem Kahveci[1] Kıvanç Dinçer[2][0000-0002-4051-4968]

[1] Hacettepe University, Ankara, TR
[2] İstanbul Gelişim University, İstanbul, TR
gizemkhv@gmail.com kdincer@gelisim.edu.tr

Abstract. In this study, a new method called *PTP-Vision* was developed to support the software project managers of a private local bank in estimating the required functional test effort/duration of the developed software units. This

method allows the project manager to predict the required test run effort (hence duration) of a test analyst for an assigned software unit at the initial test stages. On the basis, the test analyst models and measures her personal test process (PTP) as described in Humphrey's *Personal Software Process* (or *PSP*) and generate a personal estimation database by analyzing these measurements. For the development and testing of this method, real data from the bank's software development projects were used. The feasibility and sensitivity of the developed method were evaluated by comparing the estimations made at the earlier stages of the test process with the actual values of test effort. It has been found that if certain parameters related to environment and application remain stable, prediction errors do not exceed 12% band, and in most cases much smaller.

Keywords: Software test estimation, Proxy-based estimation, Personal Test Process, Banking software applications.

1 Giriş

Proje yöneticileri açısından yazılım test faaliyetleri için harcanacak zaman ve eforun doğru ve tutarlı bir şekilde tahmini oldukça önemlidir. Gerçekleşen değerlerin tahmin edilen değerler ile örtüşmemesi projenin ciddi manada başarısız olmasına sebep olabilir. Örneğin geçmişte Denver Havaalanı Otomatik Bagaj Sistemi Projesi'ndeki öngörülemeyen gecikmenin maliyetinin 340 milyon doları bulduğu hesaplanmıştır [1]. Bu bağlamda, eldeki belli kaynaklar ile test için ayrılacak süre hakkında tutarlı bir tahmin yapabilme önemli yönetim işlevlerinden birisi olmasına rağmen bunu başarabilmek özellikle erken safhalarda kolay değildir. Yazılım test süresinin projenin diğer planlama parametreleriyle karmaşık alakası dolayısıyla, proje yöneticisinin karar verme mekanizmalarını destekleyecek yöntem ve araçlara ihtiyaç duyulmaktadır.

Forrester Research şirketinin dünya çapında yaptığı araştırmaya göre [2] finans sektöründeki kuruluşların yazılım giderleri diğer sektörlere göre daha fazladır, dolayısıyla test süresine yönelik etkin bir tahmin metodu bu maliyetlerin kontrolünde katkı sağlayacaktır. İş birliği yaptığımız özel bankanın yazılım merkezinde yaşanan problem proje yöneticileri tarafından test analistlerine atanan test işleri – test projesi olarak adlandırılmaktadır– için başlangıçta uzman görüşüne bağlı olarak belirlenen sürelerin gerçekçi olmaması ve sürekli uzatılmak zorunda kalınmasıdır. Bu problemin çözümüne yönelik olarak üniversite-sanayi iş birliği ile test süreci iyileştirme çalışması yapılmıştır.

Literatüre bakıldığında test süresinin tahmini için yapılmış çalışmaların bankanın probleminin çözümü için yetersiz kaldığı görülmüştür. Burada test analistinin kişisel bazda ihtiyaç duyacağı süreyi daha test analiz ve tasarım safhasında iken tutarlı bir şekilde tahmin edebilmesi ve gecikmeden ek süre gereksinimini proje yöneticisine bildirebilmesi hedeflenmiştir.

İkinci bölümde test süresi tahmini için mevcut yöntemler incelenmiştir. Üçüncü bölümde bankada uygulanan test süreci anlatılmıştır. Dördüncü bölümde Kişisel Yazılım Süreci ve Proxy Tabanlı Tahmin Yöntemi ile ilgili bilgi verilmiştir. Beşinci bölümde geliştirilen Proxy tabanlı tahmin yöntemi tanımlanmıştır. Altıncı ve son bölümde ise yapılan pilot çalışma ve çalışmanın sonuçlarına yer verilmiştir.

2 İlgili Çalışmalar

Literatürde yazılım testi eforu tahmini alanında yapılmış çalışmaların genellikle yazılım geliştirme eforu tahmin yöntemleri uyarlanarak türetilmiş olduğu görülmüştür. Jayakumar ve Abran [3] yazılım testi eforu tahmin etmeye yönelik metotları beş ana grupta toplamaktadır: Uzman görüşüne dayalı yöntemler, Analoji ve iş kırılım yöntemleri, Faktör ve Ağırlık tabanlı yöntemler, Yazılım Büyüklüğüne dayalı yöntemler, Bulanık mantık ve diğer modellere dayalı yöntemler.

M. Chemuturi [4] yazılım testi eforunu tahmin etmek için uzman görüşüne dayalı olan Delfi Metodunu test efor tahmini için uyarlamıştır. Geniş Bant Delfi ve Başparmak Kuralı da benzeri diğer yöntemlerdir.

Yapılan diğer bir çalışmada, yazılım eforu tahmini için yaygın kullanıma sahip olan İşlev Noktası Analiz Yöntemine dayalı olarak Test Noktası Analiz Yöntemi [6] geliştirilmiştir. Bu yöntemde statik test noktaları ISO 9126 [5] uyarınca belirlenir. Hesaplanan test noktalarının toplamı üretimsel ve çevresel faktörler göz önünde bulundurularak efor olarak ifade edilir.

Nageswaran Kullanım Noktaları yöntemini geliştirmiştir [7]. Kerstner ise İşlev Noktası Analiz yöntemini test eforunu tahmin etmek için uyarlamıştır [8]. Bu yöntemler gerekli özellikler tamamlanamadığında hatalı sonuçlar üretmektedir. Aranha ve Borba Test Çalıştırma Noktası yöntemini geliştirmişlerdir [9]. İhtiyaç Analizi Dokümanından ölçülerek elde edilen test büyüklüğü ve çalıştırma karmaşıklığı bilgilerine dayalı olan bir otomatik tahmin modelinin geliştirilmesi hedeflenmiştir.

Analoji Tabanlı Kestirim yöntemi ile organizasyonlarda gerçekleşmiş birbirine benzer olan projelerden elde edilen veriler vasıtasıyla yazılım testi efor tahmini yapılmaktadır [4]. Chemuturi yaptığı diğer çalışmalarda bu yönteme benzer olan Görev Tabanlı Tahmin ve Test Senaryosu Sayma Tabanlı Tahmin Yöntemlerini geliştirmiştir. Matematiksel olarak bu yöntemler ile edilen sonuçlar doğrulanabilirdir. Ancak testin ilk safhalarında büyüklük hesaplanamamaktadır ve tahmin yapma süresi uzundur [4].

Yazılım büyüklüğüne dayalı tahmin yöntemleri olan Test Büyüklüğüne Dayalı Tahmin, COCOMO ve Slim, Seer and Knowledge Plan gibi yöntemler gerçeğe yakın sonuç üretebilmekte olup ve farklı organizasyonlarda yapılan tahminlerin birbiri ile kıyaslanmasına olanak tanımaktadır [3]. Literatürde bu yöntemler dışında bulanık mantık ve yapay sinir ağları yaklaşımlarına dayalı olarak geliştirilen yöntemlere rastlanmıştır.

3 Bankaya Özel Yazılım Test Süreci

Test faaliyetleri bu çalışmanın yapıldığı bankada V Modeline uygun olarak gerçekleşmektedir. Modelin sağ tarafında yer alan aktiviteler sol tarafında yer alan aktivitelerin çıktılarını doğrulamakta veya geçerlemektedir [10]. Bankada çalışan test analistleri V modelindeki Sistem Entegrasyon Testini uygulamaktadırlar. Test analisti geliştirilen uygulama özelliklerinin kullanıcı tarafında talep edilen fonksiyonel gereksinimleri karşılayıp karşılamadığını görmek amacıyla [11] kara kutu test yaklaşımı ile fonksiyonel testler yapmaktadır. Bu testler kullanıcının taleplerine bağlı olarak uygunluk, tamlık, birlikte çalışma ve hatalı girdi kontrollerini kapsamaktadır.

Bankanın test süreci dışarıdan alınan bir danışmanlık sonucunda tanımlanan sürece ve kullanılan yazılım yaşam döngüsü otomasyon araçlarına bağlı olarak [12]'ye göre tanımlanmıştır:

1. Test Planlama ve Kontrol – Proje planlaması ile aynı anda gerçekleşir ve test kapanış işlemleri tamamlanana kadar devam eder. Test Yöneticisi tarafından yürütülmekte olup, bu safhada Test Analistinin aktif rolü bulunmamaktadır.
2. Test Analizi ve Tasarımı – Yazılımcı birim testleri ile paralel yürütülür ve genel test hedefleri somut test koşullarına ve test senaryolarına dönüştürülür [13]. Test analistleri yazılım geliştirme sürecindeki temel kaynak olan İhtiyaç Analiz Dokümanlarını kullanarak test analizi ve tasarımı yaparlar. Bu dokümanlar her bir yazılım ürününe ilişkin süreçleri, bu süreçlerdeki normal, alternatif ve sıra dışı akışları, bu akışları yönlendiren iş kurallarını (hesaplama, kısıtlama, tetikleme ve şart ifadeleri), süreç rolleri ile bu rollerin sistem ile etkileşimini, süreçlerde iletişim kurulan yani entegre olunan diğer sistemlere ilişkin tüm bilgileri içeren dokümanlardır.

   Test Analistlerinin bu safhada, İhtiyaç Analiz Dokümanında yer alan girdiler, olaylar veya aksiyonlar ile bunların sonucunda oluşması beklenenlerin gerçekleştirilmesi için gerekli adımların açıklandığı dokümantasyonu [15] hazırlamaları gerekmektedir. Bir test senaryosu aşağıdaki bilgileri içermelidir:
   - Seviye: Test senaryosunu önceliklendirmek için kullanılan değer. Örneğin 7.4.
   - Test Adımı: Test senaryosunun gerçekleştirilmesi için gerekli adımlar. Test nesnesinin analizine, özelliklerine, davranışına ve yazılımın yapısına dayanmalıdır. Örneğin, Kullanıcıya DYBDXXX yetkisi verilmez + Birimi ilgili işlem referansını almaya yetkili olur + Yeni referans alma yetkisi verilir + İşlem Tipi Tanımlama Ekranında tanımlı olan bir referansı alır.
   - Doğruluk Kriteri: Bir test nesnesinin (fonksiyon) veya özelliğin başarılı veya başarısız olup olmadığını belirlemek için kullanılan karar verme kurallarıdır. Örneğin, B. D. Ekranı / D. R. İşlemleri referansı alamadığı görülür.
   - Case Özelliği: Pozitif veya negatif olarak iki değerden birini alabilir. Pozitif olması durumunda test senaryosu sonunda başarılı bir işlem gerçekleşmiş, negatif olması durumunda ise başarısız bir işlem gerçekleşmiş olması beklenir. Örneğin, Pozitif.
3. Test Uyarlama ve Yürütme/Çalıştırma – Yazılımcının birim testlerini tamamlanması ile başlar ve test çıkış kriterleri karşılanana kadar devam eder. Test Uyarlaması testlerin çalıştırılmaya hazırlanmasını, test verilerinin ve test senaryosunun yürütülmeye başlayabilmesi için test ortamının nihai hale getirilmesini, test senaryolarını desteklemek için test verisi belirlenmesini ve kaynak tahsisini içeren test çalıştırma çizelgesinin hazırlanmasını kapsar [13]. Testin Çalıştırılması safhasında, her bir test senaryosunun içerdiği test adımlarına karşılık gelen doğruluk kriterinin sağlandığı teyit edilir. Her bir test senaryosu çalıştırılmasında görsel olarak elde edilen sonuçlar (örneğin hata mesajları, *proxy* yanıtları) ve her bir çıkış değerinin (örneğin reel bir sayı) bulunduğu yer, başarılı veya başarısız sonuçlar için ekran görüntüsü ile birlikte, kayıt altına alınır [15]. Başarısız sonuçlar, gereksinimden, tasarımdan, kullanıcı dokümanından, standartlardan, beklenti, tecrübe veya algıdan sapma durumudur. Bankada testlerde tespit edilen başarısız sonuçlar *bulgu* olarak adlandırılmaktadır. Bankacılık alanında yaygın olarak teste gelen yazılımcı çıktısı *test nesnesi* (*test object*)

olarak ifade edilmektedir [16]. Bu çalışmada test nesnesi, test çalıştırma safhasında test analistine gelen kullanılabilir çalışan bir yazılım anlamına gelmektedir.

4. Raporlama ve Test Kapanış Faaliyetleri – Sistem test çıkış kriterleri karşılandıktan ve yürütmesinin tamamlandığı beyan edildikten sonra gerçekleştirilir. Ancak bazı durumlarda, kullanıcı kabul testi bitene ve tüm proje faaliyetleri son bulana kadar gecikebilir. Ürünün ihtiyaçlara uygun hale gelmesi, tespit edilen bulguların giderilmesi, gerçek ortama alındığında risk içermemesi halinde test planındaki test çıkış kriterlerine de uygunluğu kontrol edilerek Test Kapanış Raporu hazırlanır ve test sonlandırılır. Test Kapanış Raporu gerçek ortama alınacak uygulamadaki hataların ve yazıların test senaryolarının durumlarını gösteren rapordur. Senaryoların kaç tanesinin çalıştırıldığının bilgisi ve grafiğini içerir. Çözülmemiş bulguların riski hakkında bilgi verilir. Bulguların kaç tanesi kapatıldı vb. bilgisi ve grafiğini içerir.

4 PSP ve Proxy Yöntemi

Yazılım geliştirme projelerinde çalışan kişilerin yetenek ve çalışma disiplinleri kaliteli ürün ortaya koyabilmek için büyük önem arz etmektedir. Watts S. Humphrey, PSP (*Personel Software Process* - Bireysel Yazılım Geliştirme Süreci) konusundaki çalışmaları ile kişisel çalışma sonuçlarını iyileştirmek için tanımlanmış bir süreci izleyerek disiplin içinde yazılım geliştirmeyi sağlayan bir çerçeve sunar [17]. Kişisel disiplini ve performansı artırmak için PSP kişilere sürdürülebilir, kesin ve açık plan yapma; çalışma performansını ölçme ve izleme; ölçülen performansı artırıcı işlemler yapma; en az hata ile çalışma gibi yetenekleri kazandırmayı hedefler.

PSP kullanan yazılım geliştiricileri bireysel yazılım geliştirme süreçlerinin performansını topladıkları ölçümlere (büyüklük, geliştirme süresi, hata sayısı gibi) dayalı olarak analiz edebilirler. Buna bağlı olarak kendi geliştirme süreçlerini iyileştirebilir ve geliştirme işleriyle ilgili Proxy tabanlı kestirim yöntemi (PROBE - Proxy Based Estimating Method) ile hassas tahminler yapabilirler [17]. PROBE, büyüklüğü tarif edebilmek için kullanılan vekil nesnelere (proxy) dayalı olarak ürünü küçük parçalara ayırarak geliştirme sürelerini öngörmeyi amaçlayan bir tahmin yöntemidir. PSP yöntemini benimseyen yazılım geliştiricileri ürünlerini tip ve büyüklük olarak küçük parçalara ayırarak kategorize ederler. Her bir parça için geçmişe yönelik ölçümlemeler yaparak tahmin veritabanını oluştururlar. Yeni bir tahmin yapılması gerektiği zaman, yazılım geliştiricisi oluşturmuş olduğu tahmin veritabanında yer alan kategorilere göre geliştireceği ürünü küçük parçalara böler ve ilgili parça için tahmin veritabanından büyüklüğünü seçerek tahmin yapar. Tahminlerin iyileştirilmesine yönelik olarak istatistik yöntemleri kullanılmaktadır.

5 PTP-Vision Yöntemi

5.1 Kişisel Test Sürecindeki Temel Parametreler

Bu çalışmada bankada çalışan bir test analistinin kişisel test çalıştırma süresinin testin analiz ve tasarım safhasında iken öngörülebilmesi (tahmin edilebilmesi) hedeflenmektedir. Kişisel Yazılım Süreci (PSP) ve PROBE Yöntemi test sürecine uyarlanarak yeni bir kişisel test çalıştırma süresi tahmin metodu oluşturulmuştur.

Ölçümlerin sağlıklı yapılabilmesi için PSP'nin yazılım geliştirme süreci için önermekte olduğu Süre Kayıt Etme Formu (*Time Recording Log*) test sürecine uyarlanmıştır. Test Analisti bu formu kullanarak her bir bileşenle ilgili harcadığı süreleri kendisine atanan test projeleri için kayıt altına almaktadır.

Test analistlerinin test çalıştırma safhasındaki faaliyetleri izlenmiş ve her bir test projesinde toplam sürenin/eforun aşağıdaki faaliyetlerden/sürelerden oluştuğu tespit edilmiştir:

- İdeal şartlarda test çalıştırma süresi – Test analistinin test nesnesine ait test için bulgu çıkmadığı ve bilgi gereksinimi olmadığı bir durumda harcadığı süredir.
- Test çalıştırma süresi – Bir test senaryosunun çalıştırıldığı toplam süredir. Yazılımcı tarafından bulgunun düzeltilme süresini ve geri gönderilmesi akabinde yeniden test edilme süresini de içerisinde barındırmaktadır Bu süre, Test Süresi Kayıt Formunda <Test> olarak işlenmektedir. Bir bulgu tespit edildiği zaman testin doğruluğundan emin olmak için test dokümanlarının yeniden incelenmesi, bulguların raporlanması, bilgi alışverişi ve mailleşmeler de – test nesnesine ilişkin bilgi edinme gereksinimi çıkması durumu olarak adlandırılmaktadır.
- Bulgu düzeltme süresi – Bulgu çıkması durumunda yazılımcının ilgili test senaryosunda çıkan bulguya ne kadar zamanda döndüğünü gösteren süredir. Bu süre bulgu raporlandıktan yani yazılımcının eline ulaştıktan sonra başlatılmaktadır. Bu süre, Test Süresi Kayıt Formuna <Bulgu> olarak işlenmektedir.
- Yeniden test etme süresi – Düzeltilen bulgu ile ilgili senaryonun yeniden test edildiği süredir. Bu süre, Test Süresi Kayıt Formuna <Re-test> olarak işlenmektedir.
- Kayıplar – Test çalıştırılırken yazılım ve analist ekipleri ile yapılan bilgi alışverişleri, mailleşmeler, bulgu raporlanması kayıp süre olarak değerlendirilmektedir.

Bu çalışmada her bir projenin tek bir yazılımcısı olduğu varsayılmaktadır. Test Süresi Kayıt Formu her bir yazılımcı ve proje için ayrı tutulmakta ve her bir test senaryosu için süreler ilgili forma kayıt edilmektedir.

5.2 İdeal Şartlarda Test Çalıştırma Süresinin Tahmini

Bu sürenin tahmini için Humphrey'in PROBE yöntemi [17] test sürecine uyarlanmıştır. Öncelikle test senaryolarının test sürecini temsil edebilecek belirli özellikleri [14] sağlayan uygun vekil nesneler olacağına karar verilmiştir. Yani bir test senaryosunun büyüklüğü ile bir test senaryosunun dakika olarak çalıştırılma süresi arasında bir ilişki kurulur.

Tablo 1. Tahmin Veritabanı (Her bir senaryonun çalışma süresi (dakika))

Kategori	S	M	L
Eşleşme (MK)	8,83	20,38	31,93
Birlikte Çalışabilirlik (BC)	1,47	13,83	26,18
Ekran Tasarımı (ET)	1,13	2,38	5,01
Ekran Fonksiyonları (EF)	1,08	3,07	8,72
Çıktıların Kontrolü (ÇK)	1,67	5,99	10,30
Uyarı ve Hata Mesajları (UHM)	5,36	10,54	15,71

Her bir test senaryosunun aşağıda verilen kategorilerden sadece birine girecek şekilde ayrıntılı olarak tasarlandığı varsayılmaktadır:

1. Eşleşme (MK)
2. Birlikte Çalışabilirlik (BC)
3. Ekran Tasarımı (ET)
4. Ekran Fonksiyonları (EF)
5. Çıktıların Kontrolü (CK)
6. Uyarı ve Hata Mesajları (UHM)

Test Süresi Kayıt Formu kullanılarak 18 projeye ilişkin 1598 adet test senaryosunun (126 adet Eşleşme, 560 adet Birlikte Çalışabilirlik, 86 adet Ekran Tasarımı, 244 adet Ekran Fonksiyonu, 91 adet Çıktıların Kontrolü, 222 adet Uyarı ve Hata Mesajları) çalıştırma süreleri toplanmıştır. Akabinde her bir kategoride yer alan ölçüm sonuçları tespit edilmiştir. Kategori bazında toplanan ölçüm sonuçlarının dağılımı analiz edilerek test senaryo büyüklüklerinin Küçük (**S**), Orta (**M**) ve Büyük (**L**) olmak üzere 3 kategoriye ayrılmasına karar verilmiştir.

Toplanan ölçüm sonuçlarının istatistiksel ortalama değer ve standart sapmasının hesaplanması vasıtası ile her bir kategoride yer alan Tablo 1'de verilen Tahmin Veritabanı elde edilmiştir.

Geçmişe yönelik olarak toplanmış verilerin analiz edilmesi ile elde edilen Tahmin Veritabanı kullanılarak, yeni gelen bir projenin İdeal Şartlarda Test Çalıştırma Süresinin (P) tahmin edilebilmesi hedeflenmektedir. Bu doğrultuda her bir (S) senaryosunun tipi (x) ve büyüklük kategorisi (y) belirlenir, hangi kategoriye girdiği belirlenen senaryonun tahmini çalıştırılma süresi Tahmin Veritabanından bulunur. Bu işlem projedeki tüm senaryolar (n adet senaryo) için tekrarlanır. İdeal Şartlarda Test Çalıştırma Süresi (P) aşağıdaki gibi ifade edilmektedir:

$$P = \sum_{k=0}^{n} S(x, y)$$

Test çalıştırma süresi test senaryoları hazırlandıktan sonra tahmin edilebileceği gibi bazı durumlarda ise test senaryoları hazırlanmasının bir adım öncesinde yani ihtiyaç analiz dokümanı incelenirken öngörülmesi ihtiyacı doğmaktadır. Bu durumlarda test analisti ihtiyaç analiz dokümanını incelerken kontrolü yapılacak durumları imgeleyerek de tahmin yapabilmektedir. İmgeleyerek yapılan tahminlerde de test senaryoları üzerinden tahmin yaparken izlenen aynı adımlar takip edilmektedir.

Tablo 2. Proje Bazında Bulgu Adetleri ve Ortalama Bulgu Düzeltme Süreleri

Proje	Yazılımcı	Bulgu Adedi	Bulgu Yoğunluğu (%)	Bulgu Düzeltme Süresi (s)	Proje	Yazılımcı	Bulgu Adedi	Bulgu Yoğunluğu (%)	Bulgu Düzeltme Süresi (s)
Proje 1	1	72	79,1	2	Proje 10	5	11	11,1	9
Proje 2	1	8	20,5	3	Proje 11	6	8	28,6	10
Proje 3	1	1	11,1	2	Proje 12	4	13	21,7	1
Proje 4	2	1	0,9	1	Proje 13	6	4	19,0	3
Proje 5	3	35	18,3	1	Proje 14	6	6	8,5	2
Proje 6	2	1	5,3	1	Proje 15	6	2	2,8	5
Proje 7	3	3	20,0	6	Proje 16	5	3	10,3	6
Proje 8	3	5	17,9	2	Proje 17	6	6	8,5	4
Proje 9	4	42	13,8	2	Proje 18	7	2	2,9	4

5.3 Test Çalıştırma Süresinin Tahmini

Test çalıştırma süresi, yazılımcı tarafından bulgunun düzeltilme süresini ve geri gönderilmesi akabinde yeniden test edilme süresini de içerisinde barındırmaktadır.

Tespit edilen bulguların düzeltme sürelerinin test çalıştırma süresine etkisinin tespit edilebilmesi için, bu çalışmada 18 tane test projesine ait tespit edilen bulguların düzeltme süreleri kullanılmıştır (Tablo 2.) Bahsi geçen her bir projenin tek bir yazılımcısı vardır. Geliştirilmekte olan model için de her bir projenin tek bir yazılımcısı olduğu varsayımında bulunulmuştur. Çalışmada 18 proje için 7 farklı yazılımcı ile çalışılmıştır.

Gözlemlenen değerlerden, yazılımcıların Tablo 3'deki bulgu yoğunlukları ve ortalama bulguya dönüş süreleri çıkarılmıştır. Bulgular düzeltilip test analistine döndükten sonra yapılan düzeltmelerin başarısını doğrulamak için ilgili test senaryolarının tekrardan çalıştırılması (re-test) gerekmektedir. Yazılımcının bulgu yoğunluğu oranında, yeniden test etme süresine ihtiyaç duyduğu varsayılmıştır. Düzenleme yapıldıktan sonra koşulan test senaryolarından da m% oranında bulgu çıkabileceği ve bulgu çıkma oranının 0.001'e yakınsayana kadar t iterasyon boyunca gerçekleyebileceği farz edilmiştir.

$$\text{Yeniden Test Etme Süresi (R)} = \sum_{k=0}^{t} \lambda * P * \left(\frac{m}{100}\right)^{k}$$

Çalışmada kayıt altına alınan senaryo başına yeniden test etme adetleri proje bazında değerlendirilerek, yapılmakta olan çalışmada düzeltme yapılan senaryolardan %10 oranında (m=10) yeniden test etme ihtiyacı tespit edilmiştir.

Yukarıda bahsedilen sürelere ek olarak, Kayıplar olarak tabir edilen sürelerin Test Çalıştırma Süresine katkısı için sabit bir değer (K) vermek yerine, ideal şartlarda test çalıştırma süreleri dikkate alınmıştır. Bu çerçevede, ideal şartlardaki test çalıştırma sürelerinin [P] dağılımına göre kategoriye ayrılmış ve her bir kategori için K değerinin katkısı olmadan hesaplanan test yürütme süresi [P+B+R] ve Gerçek Test Yürütme sürelerinden de K değerleri hesaplanmıştır.

Yapılan çalışmada, Tablo 2'de verilmekte olan 18 proje için ideal şartlardaki test çalıştırma sürelerinin dağılımı incelendiğinde üç gruba ayrılabileceği öngörülmüştür.

Tablo 3. Yazılımcı Bulgu yoğunluğu ve ortalama bulgu düzeltme süresi

Yazılımcı	Bulgu Yoğunluğu (%)	Bulguya Dönüş Süresi (saat)
Yazılımcı-1	36,91	2
Yazılımcı-2	3,10	1
Yazılımcı-3	18,73	3
Yazılımcı-4	17,74	2
Yazılımcı-5	10,73	8
Yazılımcı-6	13,47	5
Yazılımcı-7	2,86	4

Bu doğrultuda elde edilen K değerleri aşağıdaki gibidir;

- K=01,29 saat eğer P < 3,50 saat
- K=14,57 saat eğer 3,50 < P < 13,28 saat
- K=27,84 saat eğer P > 13,28 saat

K değerleri 18 proje için elde edilmiş değerlerdir, toplanan verinin artması ile bu değer değişecek olup iyileşeceği düşünülmektedir.

Tablo 4'de Gerçek Test Yürütme Süresi, test analistinin Bankacılık Veritabanına her bir yazılım testi projesi için girmiş olduğu aktivite bilgilerinden alınmaktadır. Söz konusu süre sadece test faaliyetlerinin yapıldığı zamanı yansıtmaktadır. Proje sürecinde yer alan test analistinin eğitimlerini, izinlerini, molalarını vb. içermemektedir.

Yukarıda bahsedildiği gibi birbirini takip eden adımlardan oluşan ve paralelde başka bir iş için efor sarf edilmediği varsayıldığında, Tahmini Test Çalıştırma Süresi (T) aşağıdaki gibi elde edilmektedir:

$$T = \sum_{k=0}^{n} S(x,y) + \lambda * q * n + \sum_{k=0}^{t} \lambda * P * \left(\frac{m}{100}\right)^{k} + K$$

Tablo 4'de, geçmişe yönelik ölçümlerin elde ettiğimiz modele yerleştirilmesi sonucunda her bir proje için elde edilen Tahmini Test Çalıştırma Süreleri verilmektedir. Tabloda verilmekte olan sonuçlar analiz edildiğinde tahmin edilen ve gerçek süreler arasında genelde doğrusal bir ilişkinin olduğu görülmektedir (Şekil 1.) 11 No'lu proje değerlendirme dışı tutulduğunda tahmini ve gerçek süreler arasındaki ortalama bağıl hata %6,27 olarak bulunmuştur. Bu çerçevede bakıldığında, incelenen çalışma için model üzerinde oluşturulan tahminlerin gerçek süreler ile örtüştüğü gözlenmektedir.

Proje 11'in sıradışı durumu ayrıca incelenmiştir. Yazılımcı bulguları geç düzelttiği için test analistinin projeye ara verdiği ve başka bir iş aldığı anlaşılmıştır. Geliştirilen modelde test analistinin sadece tek bir proje ile ilgilendiği varsayımında bulunulmasından dolayı K değeri eklenmeden hesaplanan test çalıştırma süresinin, gerçek test çalıştırma süresinden büyük olduğu gözlenmektedir. Yazılımcının bulguya dönüş süresinin ortalamanın çok dışında olmasının hatalara yol açtığı sonucuna varılmıştır.

Table 4. Tahmini ve Gerçek Test Çalıştırma Süreleri (saat)

Proje Adı	İdeal Şartlarda Gerçek Test Çalıştırma Süresi [P]	Hesaplanan Test Çalıştırma Süresi (Kayıp Süre Eklenmeden) [P + B + R]	Tahmini Test Çalıştırma Süresi (Kayıp Süre Eklenerek) [T]	Gerçek Test Çalıştırma Süresi	Hata Oranı (%)	Yönetici Tarafından Atanmış Test Süresi
Proje-01	11	163,70	178,33	180	3,01	20
Proje-02	5	30,03	44,65	36	3,11	10
Proje-03	1	3,11	4,46	8	0,28	10
Proje-04	12	13,11	27,74	32	1,36	40
Proje-05	30	70,50	98,40	120	25,92	40
Proje-06	3	4,16	5,50	7	0,10	8
Proje-07	3	21,60	22,94	24	0,25	8
Proje-08	2	12,36	13,70	16	0,37	8
Proje-09	36	124,97	152,87	160	11,40	80
Proje-10	17	117,89	145,79	128	22,77	40
Proje-11	6	87,71	*	38	*	32
Proje-12	14	30,03	57,93	64	3,88	40
Proje-13	10	23,90	38,53	35	1,23	16
Proje-14	22	35,86	63,76	53	5,70	16
Proje-15	19	29,54	57,44	41	6,74	16
Proje-16	14	33,45	61,35	37	9,01	16
Proje-17	22	47,86	75,76	56	11,07	16
Proje-18	12	20,34	34,97	34	0,33	-

6 Pilot Çalışma

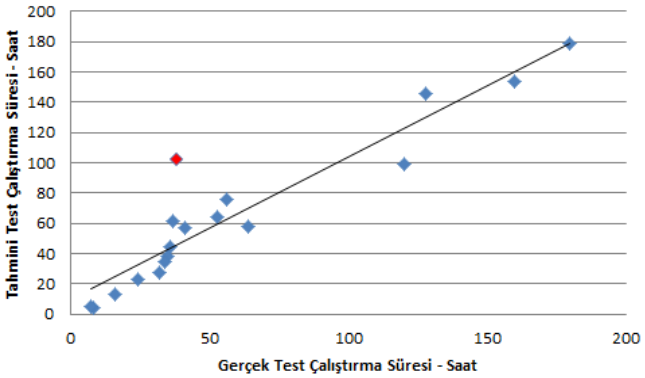
Burada bölümde, geliştirilen yöntemin önceden ölçülen değerler olmadan, test analistine atanan yeni bir proje için ileri yönelik tahmin yapılarak çalışabilirliği değerlendirilecektir.

Pilot çalışmaya konu olan test projesi; <E-posta Verilerinin Raporlanabilir Ortama Taşınması Aktivitesi> testinin gerçekleştirilmesidir. Yazılım proje kapsamında; E-posta gönderimlerinin banka çalışanları tarafından raporlanabilmesi amacı ile, mevcut e-posta verilerinin raporlanabilir ortama aktarılması sağlanacaktır. Test projesi kapsamında ise bu proje kapsamında tüm mevcut ve güncel e-posta verilerinin raporlanabilir ortama aktarımının doğru bir şekilde tamamlandığı test edilecektir.

Testi yapılacak olan projenin gereksinimleri İhtiyaç Analizi Dokümanında aşağıdaki gibi tanımlanmıştır:

- E-posta gönderimlerinin raporlanabilir ortama taşınması için gerekli tablo yapısının ve veri taşıma yöntemlerinin geliştirilmesi gerekmektedir.
- E-posta gönderimlerine ait verilerin Mars (Bankacılık Veritabanında e-posta kayıtları ile müşteri ilişkisini sağlayan özgün bir değer) ile ilişkili olarak tutulması için Mars bilgisinin mevcut tablo yapılarına eklenmesi gerekmektedir.

Bu çerçevede bakıldığı zaman iki farklı sunucuda bulunan güncel verilerin tutulduğu 8 tabloya ve mevcut verilerin tutulduğu 5 tabloya ait toplamda 116 veri sahasının doğru şekilde aktarılması beklenmektedir. Aktarımı sağlayacak olan 5 farklı program bileşeni çalıştırılacaktır. Bu sahalardan 2 tanesi (biri güncel veri için diğeri de mevcut veri için) Mars bilgisini tutmak için kullanılacak olup kontrolleri için farklı program bileşenlerinin çalıştırılması ve farklı tabloların kontrol edilmesi gereksinimi doğmaktadır.



Şekil 1. Tahmini ve Gerçek Test Çalıştırma Süreleri

Aktarımın testi ile ilgili olarak;

- Çalıştırılacak olan 5 program bileşeninin doğru çalıştığının kontrol edilmesi için birlikte çalışabilirlik kontrolü yapılacaktır. Bu kontrollerin büyüklüğünün L(büyük) olacağı öngörülmektedir:

  5 * S(BC, L)

- 116 sahanın doğru beslendiğinin görülmesi için eşleşme kontrolü yapılacaktır, bu sahalardan iki tanesinin kontrolü için farklı tabloların da kontrolüne ihtiyaç duyulduğu için büyüklüğünün L (büyük) olacağı öngörülmektedir. Kalan sahalardan 44 tanesi için büyüklüğü M (orta) olacağı düşünülmektedir, geriye kalan 70 sahanın kontrolünün ise S (küçük) büyüklüğünde olması beklenmektedir:

  2 * S(MK, L), 44 * S(MK, M), 70 * S(MK, S)

Toplamda 119 kontrol yapılmakta olup buradan testin senaryo sayısının da 119 olabileceği tespit edilmiştir. Tahmin Veritabanı kullanılarak ideal şartlarda test çalıştırma süresi 26,54 saat olarak hesaplanmaktadır.

Projenin geliştiricisi Yazılımcı-7'dir. Yazılımcı-7'nin bulgu yoğunluğu ve bulgu düzeltme süresi kullanılarak, bulgu düzeltme süresi 14,06 saat olarak ve yeniden test etme süresi 54,53 saat olarak hesaplanmıştır.

Elde edilen değerlerin toplanması ile Test Analistinin bahsedilen projeyi 71,27 saatte yani günde 8 saat çalışarak yaklaşık 9 iş gününde bitirmesi öngörülmektedir. Test Analistinin Bankacılık Veritabanına girmiş olduğu gerçekleşen aktivite süreleri incelendiğinde projeyi toplam 84 saatte ve 10,5 iş gününde bitirmiş olduğu görülmüştür.

Pilot çalışmada test çalıştırma süresinin %84,85 doğrulukta tahmin edilebildiği gözlemlenmiştir. Her yeni projenin sağladığı veriler sayesinde gelişme yaşayan Tahmin Veritabanının sağlam bir temele oturması beklenmektedir. Dolayısı ile test çalıştırma süresi tahminindeki hatanın azalacağı düşünülmektedir.

7 Sonuçlar

Bu çalışma kapsamında Watts Humphrey'in PSP ve PROXY tahmin yönteminden esinlenilmiştir. Özel bir bankada çalışan test analistlerinin kişisel test süreci modellenmiş ve geçmiş test projelerinden toplanan test süreci verilerinden bir tahmin veritabanı oluşturulmuştur. Geliştirilen proxy tabanlı PTP-Vision yöntemi pilot projeye uygulanarak test çalıştırma süresi tahmin edilmiştir. Veritabanının oluşturulması sırasında ve pilot çalışmadan elde edilen sonuçlar tahmin ve gerçekleşme süresinin büyük oranda tutarlı olması (%84,85), geliştirilen yöntemin belirli bir alanda belirli bir yazılımcı kümesi ile çalışan ve test proje büyüklükleri birbirine yakın olan durumlarda bir test analisti için uygulanabilir ve gerçekçi olduğunu göstermektedir.

Test analistinin kişisel süreci için model üzerinden yapacağı tahminlerin gerçek sonuçlar ile örtüşmesi beklenmektedir. Ancak sonuçların her duruma genelleştirilebilmesi şu an için mümkün değildir. Yukarıda belirtildiği gibi tahmin veritabanında yer alan değerler o ana kadar toplanan değerlerden oluşmakta olup, nihai değerler değildir. Yeni projelere ait ölçüm sonuçlarının toplanması durumunda bu değerler değişecektir. Şu an için her yeni proje için elde edilen ölçüm sonuçlarının daha iyi tahmin sonuçları verip veremeyeceğine yönelik bir hata analizi yapılmamıştır. Bunun için çok daha uzun bir süre ve banka yazılım merkezi genelinde bir çalışmanın yapılmasına ihtiyaç vardır. Ancak eldeki bulgulara dayalı olarak ölçüm sonuçları çoğaldıkça Kişisel Tahmin Veritabanının sağlamlaşacağı ve tutarlı tahminler vereceği beklenmektedir.

Bankacılık yazılımları sektöründe zaman önemli bir kısıt olduğu için yöntemin farklı alanda çalışan (mobil, özel projeler, nakit yönetimi, krediler vb.) birçok test analistine yaygınlaştırılması aşamasında test analistlerinin kendi süreçlerini izlemeleri ve verilerini bizim gibi kağıt ortamında kayıt altına almaları zor olacaktır. PSP'de veri toplamayı kolaylaştıran bir kısım araçlar [18] olduğu gibi ileride bu yöntemin uygulaması için de benzer araçlar geliştirilebilir.

Çalışmanın devamında genel istatistik yöntemleri yerine Geri Yayılma Algoritması (*Backpropagation*) kullanılarak ölçümlerin Tahmin Veritabanını beslemesinin sağlanması amaçlanmaktadır. Geri yayılım algoritması hataları geriye doğru çıkıştan girişe azaltmaya çalışmaktadır. Böylelikle yeni projeye ait ölçüm sonuçları ile Tahmin Veritabanının iyileşip iyileşmediği tespit edilebilecektir. İyileştiren değerlerin Tahmin Veritabanına beslenmesi yolu ile veritabanının sağlamlaştırılması hedeflenmektedir.

Kaynakça

1. Akagündüz, S., Kurnaz, S. and Sari, M.: Yazılım Proje Yönetiminde Proje Başarısını Getiren Faktörler (Factors that Make the Success of the Project in Software Project Management.) In: Akademik Bilişim 2013 (2013)
2. Mai, H.: IT in Banks : What does it cost ? (2012).
3. Jayakumar, K.R., Abran, A.: A Survey of Software Test Estimation Techniques. J. Softw. Eng. Appl. 6, 47–52 (2013).
4. Chemuturi, M.: Software Estimation Best Practices, Tools and Techniques: A Complete Guide for Software Project Estimators. Florida (2009).
5. ISO/IEC: International Standard ISO/IEC 9126-1 Software engineering - Product quality - Part 1: Quality model. (2001).

6. Veenendaal, van EPWM (Erik); Dekkers, T.: Test Point Analysis : a Method for Test Estimation. Proj. Control Softw. Qual. 1–16 (1999).
7. Nageswaran, S.: Test Effort Estimation Using Use Case Points, Use Case Points. Quality Week 2001, San Francisco, pp. 1–6 (2001)
8. Kerstner, M.: Software test effort estimation. SIGSOFT Softw. Eng. Notes (2011)
9. Aranha, E., Borba, P.: Estimating manual test execution effort and capacity based on execution points. Int. J. Comput. Appl. 31, 167–172 (2009).
10. Firesmith, D.: SEI Blog: Using V Models for Testing. Carnegie Mellon University (2013)
11. SWEBOK. Guide to the Software Engineering Body of Knowledge, 2004 Version (2004)
12. Chan, A., Wei, J.J., Boyer, J., Guo, P., Wang, H.Y., Liu, H.L.: IBM Business Process Manager testing methodology , Part 1 : General testing guidelines. (2014).
13. ISTQB. Standard Glossary of Terms Used in Software Testing. Version 3.1 (March 2016)
14. Humphrey, W.: A Discipline for Software Engineering. Addison Wesley, Boston (1995)
15. IEEE, "IEEE Std. 829-2008: Standard for Software and System Test Documentation. (2008)
16. Hass, A.M.J.: Guide to Advanced Software Testing. Artech House (2008).
17. Humphrey, W.: PSP: A Self-Improvement Process for Software Engineers. SEI Series in Software Engineering, Edition 1 (2005)
18. Çalışkan, S., Çetinkaya, E., Dinçer, K., Yılmaz, E., Çakıcı, H.: PSP Eğitimi için Kullanıcı Dostu bir Süreç Yönetim Aracı Geliştirme Denemesi (A Process Management Tool Development Trial for PSP Training). In: Proceedings of the 9th Turkish National Software Engineering Symposium (UYMS 2015). pp. 529–541 (2015).