Yazılım Kalitesini Değerlendirmek İçin Bir MDP Tabanlı Simülasyon Modeli ve Bir Örnek Uygulama

Ömer Korkmaz[1] İbrahim Akman[2]

[1]Deniz Savaş Sistemleri Grup Başkanlığı, HAVELSAN A.Ş., Ankara
[2]Bilgisayar Mühendisliği Bölümü, Atılım Üniversitesi, Ankara

[1]okorkmaz@havelsan.com.tr [2]akman@atilim.edu.tr

Özetçe

Bu makale, yazılım kalite seviyesinin, projenin erken safhalarında değerlendirilebilmesi için bir Markov Karar Süreci (MDP) tabanlı simülasyon modeli önermektedir. Önerilen yaklaşım, yazılım geliştirme sürecinin stokastik doğası üzerine kurulmuştur ve proje mimarisini, Yazılım Kalite Güvence (YKG) sisteminde belirlenen niteleme faaliyetlerini, YKG sistemini oluşturma stratejisini ve projedeki takım atama stratejisini girdi olarak almakta ve göreceli bir kalite derecesini çıktı olarak vermektedir. Önerilen yaklaşım, literatürden seçilen bir örnek proje üzerinde uygulanmıştır. Sonuçlar, proje geliştirme esnasında izlenmesi muhtemel stratejilerin, kalite, maliyet ve zaman açısından farklılıklar gösterdiğini ortaya çıkarmıştır.

1. Giriş

Yazılım, her çeşit sistemlerde önemi gittikçe artan bir unsurdur. Şu anki ve gelecekteki sistemlerin yetenekleri, büyük ölçüde yazılımlarının performansına bağlıdır [1].

Ancak, bir yazılım ürününün kalitesini değerlendirmek ve yüksek tutmak, diğer endüstriyel ürünlere göre daha zordur. Ayrıca, yazılım geliştirme ve bakım işi, hata yapmaya eğilimi olan, zaman alıcı ve karmaşık bir aktivitedir. Bu durum, yazılım kalitesini önemli ölçüde etkilemektedir. Yazılım kalitesini etkileyen önemli unsurlardan bazıları, hatalı gereksinim tanımları, müşteri-geliştirici arasındaki iletişim yetersizliği, tasarım ve kod hataları, test sürecinin yetersizliği, usul ve dokümantasyon hataları olarak sıralanabilir [2].

İstatistikler, yazılım projelerinin başarıyla ya da yüksek kalite seviyesinde tamamlanma oranının, diğer endüstrilerdekine nazaran çok daha düşük olduğunu göstermektedir. Dahası, literatür, yazılım hatalarını düzeltme maliyetinin ve süresinin, yazılım geliştirme fazları ilerledikçe zamana göre katlanarak arttığını belirtmektedir [3, 4].

Yazılım endüstrisinde ve akademik çevrelerde hala yazılım kalitesinin tanımı üzerinde bir fikir birliği sağlanamamıştır. Bundan dolayı literatür, tatmin edici yazılım kalite seviyesine ulaşmak için farklı kalite süreç ve standartları önermektedir.

Balan'ın önerdiği YKG modeli [5], düşük-orta olgunluk seviyesine sahip organizasyonlarda, süreç geliştirmede KG'nin rollerini açıklayarak yazılım geliştirme sürecini daha görünür kılmayı amaçlamaktadır. Drabick'in süreç modeli [9], yazılım kalitesini yükseltmek amacıyla YK Mühendisliği süreci oluşturmada bir temel olarak kullanılabilir. Prabhakar makalesinde [10] bazı yararlı KG pratiklerini açıklamıştır. Tamres [11] daha iyi bir kalite seviyesi elde etmek için hataları önleyecek veya üründeki hataları tespit ederek ayıklayacak bazı anahtar süreçler önermiştir. Ntourntoufis [12] yazılım hatalarının oluşmasını önlemek ve oluşan hataları tespit etmek için kullanışlı bir araç olarak, yazılım geliştirme bilgi sistemini (DIS) sunmuştur. Harris [13] yazılım kalite seviyesini yüksek tutmak için bir doğrulama ve geçerleme (V&V) metodolojisi önermiştir. Padberg'in sunduğu model [6], yazılım projeleri için optimal zaman çizelgesi (schedule) elde etmeyi amaçlamaktadır. Bu model, yazılım ürününde oluşan tüm hataların tespit edilebildiğini ve tespit edilen tüm hataların üründen ayıklanabildiğini varsaymaktadır. Keeni, Srivastava and Sharma [14] tarafından geliştirilen matematiksel model, yazılım teftişi (inspection) üzerine odaklanmıştır. Fakat model, YKG sistemi ve yazılım geliştirme süreci için tam bir sistem tanımlamamaktadır.

Yukarıda bahsi geçen yaklaşımların çoğu, yazılım kalitesini yükseltmek için çeşitli metodlar önermelerine rağmen, kalite üzerinde istenen yargıyı sağlamak için analizcinin önsezisine ve tecrübesine ihtiyaç duymaktadır [15].

Bu çalışmada, daha önceden önerilen MDP tabanlı modelin [16] bir uygulaması sunulmuştur. MDP, içinde bir sistemin uzun dönem davranışlarını en uygun hale getirebilmek (optimization) için, sistemi tanımlayan bir matematiksel modeldir. Önerilen model, yazılım geliştirme sürecinin stokastik doğasını dikkate alarak, yazılım kalite seviyesini, projenin erken safhalarında değerlendirmeyi amaçlamaktadır. Model, yeni (novel) bir model olup, projenin YKG sistemini oluşturma stratejisini, projenin mimari yapısını, projenin parçalarına takım atama stratejisini ve proje takımlarıyla YKG unsurlarının (SQA components) karakteristiğini girdi olarak almakta ve göreceli bir optimal kalite derecesini çıktı olarak vermektedir. Önerilen yaklaşım parametrize edildiğinden, projenin sonraki safhaları için farklı teorik performans hedeflerini test edebilme avantajına sahiptir. Diğer bir önemli avantajı ise kullanıcının karar alırken farklı iş yükü ve konfigürasyonları dikkate almasına imkan vermesidir.

Bu çalışmada sunulan uygulama, proje yöneticileri tarafından bir yazılım projesinin kalite seviyesini, geliştirme sürecinin erken safhalarında değerlendirmek amacıyla kullanılabilir. Uygulama, yazılım geliştirmede kullanılabilecek farklı politikaları denemeyi kolaylaştırmaktadır. Yönetici, izlenecek politikayı değiştirdiği zaman, kalite derecesinin nasıl değiştiğini çabucak görebileceği için, farklı politikaları karşılaştırarak en iyisini seçebilir.

Bu makalenin geri kalanı şu şekilde düzenlenmiştir: Bir sonraki bölüm, [16]'da sunulan *niteleme (qualification) model*ini ve bunun üzerine kurulan *MDP model*ini genel olarak açıklamaktadır. Sonraki bölüm, *MDP model*inden türetilen *simülasyon model*inin [16] Arena® simülasyon paket programı kullanılarak uygulanması açıklamaktadır. Uygulama, literatürden alınarak bazı eklemeler yapılan basit bir örnek proje [6] üzerinde çalıştırılmış, elde edilen sonuçlar incelenmiş ve tartışılmıştır. Sonuç ve Tavsiyeler, makalenin son iki bölümünü oluşturmaktadır.

2. Modeller ve Simülasyon

2.1. Niteleme Modeli

Bu bölümde her iki model de çeşitli varsayımlar dikkate alınarak açıklanmıştır. *Niteleme modeli*, projedeki görev atamalarını, personel ve YKG unsurlarının yetenek seviyelerini, ve hataların sebep olduğu *yeniden çalışma* (rework) işlerini temsil eden *stokastik* bir modeldir. Model stokastiktir, çünkü geliştirme fazlarındaki aktivitelerin tamamlanma süresinin ve olayların meydana gelmesinin belirsizliğini hesaba katmaktadır.

Niteleme modeli, her bir *iterasyon* bir küçük proje kabul edilerek, *Ardışık ve Artan Geliştirme Süreci* (Iterative & Incremental Development Process) için de kullanılabilir. Bu amaçla, model üzerinde bir takım eklemeler yapılarak her bir iterasyonun kalite değeri üzerinden projenin bütününün (overall) kalite değeri hesaplanabilir.

2.1.1. Proje Mimarisi

Bir proje, üst seviye tasarım çalışmaları esnasında birden fazla *parça*ya (component) bölünür. Proje parçaları büyüklüklerine ve karmaşıklık seviyelerine göre sınıflandırılabilir. Karmaşıklık seviyesinin, proje parçası üzerinde geliştirme yapılırken hata yapma olasılığının seviyesi ile orantılı olduğu varsayılabilir.

Projenin bir parçası üzerinde çalışan personel grubu, *takım* olarak adlandırılmaktadır. Model, proje takımlarının birbirinden bağımsız çalıştığını varsaymaktadır. Bir başka deyişle, bir proje parçasında bulunan bir hatanın (veya hataların), diğer proje parçalarını etkilemediği varsayılmıştır.

2.1.2. Temel Olasılıklar

Niteleme modeli olasılıksal bir modeldir, çünkü yazılım geliştirme yaşam döngüsü boyunca olaylar belirli olasılık değerleriyle meydana gelir. Bu olasılık değerlerinin tahmini zorunludur ve şirketin eski projelerine ait metriklerini tutan metrik veritabanından elde edilebilecek *istatistiksel data*lara dayandırılmalıdır. İstatistiksel datalar, proje takımlarının, şirketin eski projelerinde nasıl ilerlemeler yaptığını gösterir. Dolayısıyla, olasılıkları doğru hesaplayabilmek için yeterince büyük bir metrik veritabanına sahip olmak iyidir [7, 8].

Niteleme modeli için *temel olasılıklar* aşağıdaki gibidir:

1. Proje takımının, geliştirdiği ürün üzerinde hata yapma olasılığı,
2. Proje ürünü üzerinde uygulanan YKG unsurunun, hata bulma olasılığı,
3. Proje takımının, bulunan hataları, *yeniden çalışma* ile üründen çıkarma olasılığı.

Bu çalışmada, yazılım projesinin parçaları 9 sınıfa ayrılmıştır. Buna göre proje parçasının karakteristiği, Tablo 1'de gösterildiği gibi, parçanın tahmin edilen zorluk seviyesi ve büyüklüğüyle belirlenmiştir.

Benzer şekilde, proje takımları da 9 sınıfa ayrılmıştır. Bir takımının yetenek ve maliyet seviyesi, Tablo 2'de görüldüğü gibi, takımın üretkenlik ve hata yapma seviyeleriyle belirlenmiştir. Maliyet seviyesi, 0.80 ile 1.00 birimler arasında kademelendirilerek sıralanmıştır. Gerçek projelerde, takımlar için birim maliyet kademelendirmesi, şirketin ücret politikası dikkate alınarak yapılmalıdır. Dolayısıyla, şirketten şirkete birim maliyet kademeleri farklılık gösterebilir. Takımların yetenek seviyeleri (üretkenlik ve hata yapma seviyeleri), temel olasılıkların hesaplanmasında kullanılırken; maliyet seviyeleri (veya birim maliyetler), proje maliyetini hesaplamak için kullanılır. Dolayısıyla, temel olasılıklar, kalite seviyesi üzerinde etkiye sahipken, maliyet kalite derecesi (birim maliyet başına birim kalite değeri) üzerinde etkilidir.

Tablo 1: Yazılım proje parçalarının sınıfları

Parça Sınıf No	Tahmin Edilen Büyüklük	Tahmin edilen Zorluk Seviyesi
1	Küçük	Düşük
2	Küçük	Orta
3	Küçük	Yüksek
4	Orta	Düşük
5	Orta	Orta
6	Orta	Yüksek
7	Büyük	Düşük
8	Büyük	Orta
9	Büyük	Yüksek

Tablo 2: Yazılım proje takımlarının sınıfları

Takım Sınıf No	Üretkenlik Seviyesi	Hata Yapma Seviyesi	Maliyet Seviyesi	Birim Maliyet
1	Düşük	Düşük	7	0.850
2	Düşük	Orta	8	0.825
3	Düşük	Yüksek	9	0.800
4	Orta	Düşük	3	0.950
5	Orta	Orta	4	0.925
6	Orta	Yüksek	6	0.875
7	Yüksek	Düşük	1	1.000
8	Yüksek	Orta	2	0.975
9	Yüksek	Yüksek	5	0.900

2.1.3. Niteleme Faaliyetleri

Model, n*iteleme faaliyetleri*nin (qualification actions) sadece, bir takımın atandığı proje ürünü üzerindeki geliştirme çalışmalarını bitirdiği zaman gerçekleştirileceğini kabul etmektedir. Örneğin; gözden geçirmeler, resmî tasarım gözden geçirmeleri, testler, vb …

Modele göre proje ürünlerini geliştiren takımların aksine, niteleme faaliyetlerini yapacak takımlar baştan seçilemez. Çünkü bu takımları seçerken şunlar dikkate alınmalıdır:

1. Seçilecek takım üyesinin, üzerinde niteleme faaliyeti yapılacak ürün hakkındaki tecrübe ve bilgisi,
2. Seçilecek takım üyesinin, niteleme faaliyeti yapılacağı anda müsait olup olmaması.

Bundan dolayı model, niteleme faaliyetlerini yapan takımların yetenek seviyesinin, yazılım kalitesi üzerindeki etkisini dikkate almamaktadır. Ayrıca, bir proje parçası üzerinde yapılması gereken *yeniden çalışma*yı, proje parçasını geliştiren takımın yapacağı varsayılmıştır. Niteleme faaliyetlerinin karakteristiği, *temel olasılıklar* kullanılarak modellenmiştir.

2.1.4. Takım Atama Stratejisi

Niteleme modeli, geliştirilen projenin birden fazla parçadan oluştuğunu ve birden fazla takım tarafından geliştirildiğini varsaymaktadır. Ayrıca, bir proje parçası üzerinde en fazla bir proje takımı çalışırken, bir proje takımının en fazla bir proje parçası üzerinde çalışabileceği varsayılmıştır. Bir proje takımı, bir proje parçası üzerindeki işini bitirince, yeni bir proje parçasına atanabilir. Ama model, takımın, mevcut proje parçası üzerindeki çalışmasını tamamlamadan yeni bir proje parçasına atanmayacağını varsaymaktadır.

Modelin olasılıksal doğası, proje parçaları üzerindeki çalışmaların tamamlanma sürelerinin peşinen bilinememesine sebep olmaktadır. Bundan dolayı, bir öncelik listesinin hazırlanmasının [16] hiç bir anlamı yoktur. Bir başka deyişle, modelde gerçek *iş takvimi* (schedule), takımların atandıkları proje parçaları üzerindeki çalışmalarını bitirme sırasına bağlıdır ki; bu da şansa bağlıdır.

Model, proje takımlarının, farklı yetenek seviyelerine sahip olabileceklerini kabul etmektedir. Takımların yetenek seviyeleri, şirketin daha önceki projelerinden elde edilen metrik veritabanından türetilebilir. Buna göre, bir proje takımının yetenek seviyesi, üretkenlik ve hata yapma oranıyla belirlenebilir. Bir proje takımının yetenek seviyesi yüksek ise, ürettiği proje ürünü üzerinde hata olma olasılığı daha azdır. Proje takımlarının yetenek seviyesi, *temel olasılıklar* kullanılarak modellenmiştir. Bir diğer varsayım ise, projeye atanan takım sayısının, proje boyunca sabit kaldığıdır.

2.1.5. YKG Sistemini Oluşturma Stratejisi

Bu strateji, bir organizasyonda YKG sistemini uygularken YKG unsurlarını seçmekte izlenen yoldur. Organizasyon için oluşturulan YKG sistemi, organizasyondaki bölümler ve onların projeleri tarafından, kendi içsel dinamiklerini yansıtacak şekilde uyarlanabilir. Bunun için şunlar dikkate alınmalıdır:

1. Geliştirilen projenin karakteristiği;
2. Proje için önemli olan kalite faktörleri;
3. Bölümün kalite politikası ve amaçları; ve
4. Proje personelinin yetenek seviyesi.

2.2. MDP Modeli ve Simülasyon

*Niteleme model*ine dayandırılarak bir *MDP modeli* önerilmiştir [16]. Bu model, niteleme modelinin, sadece tasarım fazını içerecek şekilde basitleştirilmiştir. Daha sonra MDP modeli, simülasyon teknikleri kullanılarak *simülasyon model*ine dönüştürülmüştür [16].

Bu çalışmada, yazılım geliştirme sürecini, kalite bakış açısıyla modellemek için MDP tabanlı *simülasyon model*inin pratik uygulamasının detayları açıklanmış, uygulama örnek bir proje üzerinde çalıştırılmış ve simülasyonun sonuçları detaylarıyla incelenmiştir.

3. Uygulama

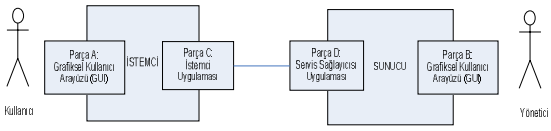
*MDP tabanlı simülasyon model*inin uygulamasını basitleştirmek için sadece aşağıdakiler hesaba katılmıştır:

1. Proje takımlarının ve YKG unsurlarının yetenek seviyeleri;
2. Proje parçalarının büyüklük-karmaşıklık seviyeleri;
3. Parçaların geliştirme sıraları;
4. Parçalarda tespit edilen hataları ayıklamak için yapılan yeniden çalışmalar; ve
5. Takımların görev atamalarında yapılan değişiklikler.

Önerilen modelin daha fazla özellik içerecek şekilde genişletilebileceğine dikkat edilmelidir. Örneğin, proje parçalarının birbirlerine bağımlı olmasını dikkate almak gibi. Önerilen MDP tabanlı simülasyon modelini, gerçek bir yazılım projesi üzerinde uygulamak, bazı zaman alıcı ön işleri tamamlamayı gerektirmektedir. Örneğin, proje takımlarının daha önce geliştirilmiş projelerdeki performanslarına göre bir metrik veritabanı yaratılmalı ve her bir takımın temel olasılık değerleriyle olasılık dağılım tabloları oluşturulmalıdır. Bundan dolayı, önerilen modelin performasını analiz etmek için literatürden alınan küçük bir farazi yazılım projesi [6] kullanılmıştır. Bu örnek projenin seçilme sebeplerinden biri de daha önce yayımlanmış bir çalışmada kullanılmış olmasıdır.

3.1. Örnek Projenin Mimarisi

Örnek proje, dört parçadan oluşan bir istemci-sunucu sistemidir [6]. İstemci, bir küçük ön yüz (proje parçası A) ve bir büyük uygulama (proje parçası C) kısmından oluşmaktadır. Sunucu, bir küçük yönetici ön yüzünden (proje parçası B) ve bir büyük uygulama çekirdeğinden (proje parçası D) oluşmaktadır. Örnek projenin blok diyagramı Şekil 1'de verilmiştir.



Şekil 1: Sistem blok diyagram

Tablo 1'e göre, proje parçası A, Sınıf 1'e (küçük-düşük); proje parçası B, Sınıf 4'e (orta-düşük); proje parçası C, Sınıf 8'e (büyük-orta) ve proje parçası D, Sınıf 9'a (büyük-yüksek) ait olacak şekilde seçilmiştir.

Örnek projede iki takım vardır: Takım 1 ve Takım 2.Tablo 2'ye göre Takım 1, Sınıf 7'ye (yüksek-düşük) ve Takım 2, Sınıf 4'e (orta-düşük) ait olacak şekilde seçilmiştir. Seçme kriterleri, Padberg'in çalışmasında [6] proje parçaları ve takımlar hakkında belirttiği varsayımlar dikkate alınarak oluşturulmuştur.

Proje parçalarının büyüklük ve zorluk seviyeleri ile proje takımlarının üretkenlik ve hata yapma seviyeleri, takip eden bölümlerde açıklanan olasılık değerleriyle ve olasılık dağılımlarıyla modellenmiştir.

3.2. Temel Olasılıklar

Temel olasılıklar, proje parçalarının, YKG unsurlarının (bu çalışmada sadece *tasarım gözden geçirme*) ve proje takımlarının karakteristiklerini yansıtmaktadır. Bir başka deyişle, temel olasılıklar, proje takımlarının proje parçalarını geliştirirken nasıl ilerlediklerini, YKG unsurlarının proje parçaları üzerinde nasıl davrandıklarını ve proje takımlarının proje parçaları üzerinde yeniden çalışma yaparken nasıl ilerlediklerini göstermektedir.

Örnek projede Padberg, temel olasılıkları hesaplamak için *iki terimli dağılımlar* (binomial distributions) kullanmayı önermiştir. Olasılıklar, iki terimli olasılık değerleri ile *ölçek faktörü* (scale factor) çarpılarak hesaplanmıştır.

Bu çalışmada, Padberg'in temel olasılıkları aşağıdaki kurallara göre uyarlanmıştır:

1. Padberg'in çalışmasındaki "bir *k takımı*nın bir *r proje parçası* üzerinde *t birim zaman* boyunca çalışarak proje parçasını bitirme olasılığı ($P(D_r^k(t))$)", bu çalışmada "bir *k takımı*nın bir *r proje parçası* üzerinde *t birim zaman* boyunca çalışarak proje parçasını hatasız bitirme olasılığı"na denk düşmektedir.
2. Padberg'in çalışmasındaki "bir *k takımı*nın bir *r proje parçası* üzerinde *t birim zaman* boyunca çalışınca bir tasarım hatası rapor etme olasılığı ($P(\varepsilon_r^k(t))$)", bu çalışmada "bir *k takımı*nın bir *r proje parçası* üzerinde *t birim zaman* boyunca çalışarak proje parçasını en az 1 hata ile bitirme olasılığı"na denk düşmektedir.

Her bir "proje takımı-proje parçası" için ve her bir "YKG unsuru-proje parçası" için, ayrı birer temel olasılık kümesi kullanılmıştır.

Kullanılan temel olasılıklar *sonlu destekli ayrık dağılım*lardır (discrete distributions with finite support). Bundan dolayı, tasarım fazı için *eğrilmiş iki terimli dağılım*lar (skewed binomial distributions) kullanılırken, tasarım gözden geçirme ve yeniden çalışma fazları için *eğrilmiş üç terimli dağılım*lar (skewed trinomial distributions) kullanılmıştır.

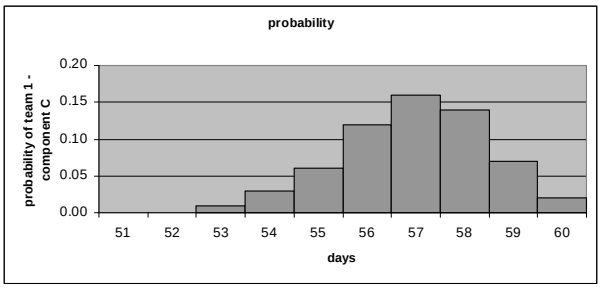
Tablo 3'te, Padberg'in çalışmasından alınan *iki terimli dağılımlar*ın ilgili parametreleri, *tepe değer*leri ve *ölçek faktörleri* listelenmiştir. Bu tablo, örnek proje için tasarım fazında proje parçası geliştirme ile ilgili temel olasılıkların hesaplanmasını göstermektedir.

Tablo 3: Proje parçası geliştirme ile ilgili temel olasılıkların hesaplanması

Olasılık	Dağılıma ait parametreler		Tepe	Ölçek
	n (hafta)	p		
$P(A_1^A(t))$	7	0.7	6	0.8
$P(B_1^A(t))$	6	0.5	4	0.2
$P(A_2^A(t))$	10	0.7	8	0.8
$P(B_2^A(t))$	10	0.5	6	0.2
$P(A_1^B(t))$	7	0.8	7	0.8
$P(B_1^B(t))$	6	0.55	4	0.2
$P(A_2^B(t))$	10	0.8	9	0.8
$P(B_2^B(t))$	10	0.55	7	0.2
$P(A_1^C(t))$	12	0.75	10	0.6
$P(B_1^C(t))$	13	0.45	7	0.4
$P(A_2^C(t))$	16	0.75	13	0.6
$P(B_2^C(t))$	20	0.45	10	0.4
$P(A_1^D(t))$	14	0.75	12	0.4
$P(B_1^D(t))$	15	0.45	8	0.6
$P(A_2^D(t))$	20	0.75	16	0.4
$P(B_2^D(t))$	26	0.45	13	0.6

Tablo 3'teki *tepe değer*inin, iki terimli dağılımda [(n+1).p+1] noktasında elde edildiğine dikkat ediniz. Tablodaki parametreler ve ölçek faktörleri, firmanın istatistiksel veritabanından oluşturulabilir.

Şekil 2, Takım 1 için proje parçası C üzerinde geliştirme yaparken zamana bağlı olarak "hiç hata yapmama olasılıklarını" göstermektedir. Şekilde, sol tarafındaki kuyruğun daha uzun olduğu görülmektedir; yani kütle sağ tarafta yoğunlaşmıştır. Dolayısıyla, grafik *soldan eğrilmiş*tir (left-skewed). Eğrilik, simetri için bir ölçüdür; daha kesin ifade ile simetrinin noksanlığının bir ölçüsüdür. Koyu gölgeli alanın değeri 0.61 olup Takım 1'in proje parçası C üzerinde çalışırken "hiç hata yapmama olasılığı" değerini ifade etmektedir.



Şekil 2: Olasılık dağılımı $P(A_1^C(\mathbf{t}))$

Aynı muhakeme, diğer temel olasılıkları hesaplarken de uygulanabilir. İlgili bilgiler Tablo 4 ve Tablo 5'te verilmiştir. Tablo 4, hata tespiti amacıyla yapılan *tasarım gözden geçirme* niteleme faaliyeti için ölçeklendirilmiş temel olasılıkları göstermektedir.

Tablo 4: Hata bulma ile ilgili temel olasılıkların hesaplanması

Olasılık	Dağılıma ait parametreler		Tepe	Ölçek
	n (gün)	p		
$P(C_{DR}^A(t))$	5	0.60	4	0.5
$P(D_{DR}^A(t))$	6	0.67	5	0.3
$P(C_{DR}^B(t))$	7	0.57	5	0.5
$P(D_{DR}^B(t))$	7	0.71	6	0.3
$P(C_{DR}^C(t))$	9	0.67	7	0.4
$P(D_{DR}^C(t))$	10	0.70	8	0.4
$P(C_{DR}^D(t))$	12	0.75	10	0.3
$P(D_{DR}^D(t))$	14	0.71	11	0.5

Tablo 5, tasarım üzerinde *yeniden çalışma* (rework) fazında *hata ayıklama* faaliyeti için ölçeklendirilmiş temel olasılıkları göstermektedir.

Tablo 5: Hata ayıklama ile ilgili temel olasılıkların hesaplanması

Olasılık	Dağılıma ait parametreler		Tepe	Ölçek
	n (gün)	p		
$P(F_1^A(t))$	10	0.70	8	0.7
$P(G_1^A(t))$	12	0.50	7	0.2
$P(F_2^A(t))$	13	0.70	10	0.7
$P(G_2^A(t))$	13	0.50	8	0.2
$P(F_1^B(t))$	12	0.67	9	0.7
$P(G_1^B(t))$	13	0.46	7	0.2
$P(F_2^B(t))$	15	0.67	11	0.7
$P(G_2^B(t))$	17	0.46	9	0.2
$P(F_1^C(t))$	18	0.78	15	0.5
$P(G_1^C(t))$	20	0.50	11	0.3
$P(F_2^C(t))$	22	0.78	18	0.5
$P(G_2^C(t))$	23	0.50	13	0.3
$P(F_1^D(t))$	22	0.77	18	0.3
$P(G_1^D(t))$	24	0.42	11	0.5
$P(F_2^D(t))$	25	0.77	22	0.3
$P(G_2^D(t))$	25	0.42	12	0.5

Tablo 4 ve Tablo 5'teki değerler de Tablo 3 oluşturulurken takip edilen Padberg'in çalışmasındaki varsayımlara göre oluşturulmuştur.

Yazılım geliştirme sürecinin olasılıksal doğasından dolayı, en iyi kalite sonucunu almak için geliştirmenin erken safhalarında hangi stratejinin seçilmesi gerektiği belirsizdir. Dolayısıyla, bu küçük örnek projeyi kullanmak, yazılım geliştirme sürecinin pek çok özelliği yansıtmaktadır.

3.3. Takım Atama Stratejileri

Yazılım geliştirme esnasında izlenen yol, proje yöneticisinin seçtiği stratejilere bağlı olup, projenin içinde bulunması muhtemel her bir *durum* (state) için yapılacak *eylem*leri (action) belirler. Bundan dolayı, izlenmesi muhtemel yolların toplam sayısı, projedeki muhtemel *durum* ve *eylem* sayısına bağlıdır.

Gerçek projeler geliştirilirken, bir takımın bir proje parçasına atanması esnasında proje takımlarının yetenek seviyesi ve proje parçalarının sınıfı dikkate alınmalıdır. Gerçek projelerde kullanılan stratejiler, hangi proje parçalarının hangi proje takımları tarafından geliştirileceğini belirler. Fakat gerçek projelerin tersine, bu simülasyon modeli, olasılıksal doğası yüzünden, bir sonraki parçanın geliştirilmesine başlamadan önce "en uygun" takımın serbest olmasını bekleyemez.

3.4. YKG Sistemini Oluşturma Stratejisi

Bu çalışmadaki MDP tabanlı simülasyon modeli, sadece yazılım geliştirme sürecinin tasarım fazını içerdiğinden, projelerdeki YKG sistemini oluşturma stratejisi, model için bir girdi değildir. Modelde kullanılan yegane YKG unsuru, *tasarın gözden geçirme* unsurudur.

3.5. Modelin Arena® Simülasyon Aracında Uygulaması

Simülasyon modelinin uygulaması, genel amaçlı grafiksel simülasyon aracı Arena® [17] kullanılarak gerçekleştirilmiştir. Arena®, iş, servis ve üretim süreçlerini modellemek ve analiz etmek için en çok tercih edilen simülasyon araçlarından birisidir [18].

A, B, C ve D proje parçalarını yaratmak için *Create Flowchart Module*; farklı varlık (entity) tiplerini (A, B, C ve D proje parçaları) tanımlamak için *Entity Data Module*; süreçleri (proje parçaları üzerinde geliştirme yapmak, niteleme faaliyetlerini gerçekleştirmek, proje parçaları üzerinde *yeniden çalışma* yapmak) temsil etmek için *Process Flowchart Module*; kaynakları (proje takımları 1 ve 2) ve onların karakteristiklerini tanımlamak için *Resource Data Module* kullanılmıştır. Varlık tiplerine (proje parçaları) ve kaynaklara (proje takımları) bağlı olarak süreç süreleri, olasılık dağılımları kullanılarak modellenmiştir. Çeşitli varlık özelliklerini (entity attribute) ve sistem değişkenlerini tanımlamak ve sistem boyunca onlara yeni değerler atamak için *Assign Flowchart Module* kullanılmıştır. Süreç modülleri tarafından gerçekleştirilen işlerin sonuçlarını incelemek için, yazı-tura atmaya dayalı "geçti" veya "kaldı" değerleriyle birlikte *Decide Flowchart Module* (2-way veya N-way) kullanılmıştır. Yazı-tura için para fırlatmanın gösterdiği davranışın, ilgili olasılık değerlerini yansıtacağına dikkat edilmelidir. Bir proje geliştirme stratejisinin toplam ödül ve maliyet değerleri gibi çeşitli çıktı değerlerini ölçmek için *Statistics Data Module* kullanılmıştır.

Tekrarlama sayısı 1000 olarak ayarlanmış ve "Statistics" ile "System" değişkenleri, her tekrarın başında yeniden sıfırlanmıştır. "Base Time Unit" olarak "days" seçilmiş ve diğer tüm "Run Setup" parametreleri için varsayılan (default) değerler kullanılmıştır.

Olasılık dağılımlarını ifade etmek için, örnek veriler üzerinde *Input Analyzer* aracı [19, 20] kullanılmıştır. Olasılık dağılımları, *Arena® Discrete Probability Distribution* ile temsil edilmiştir. Süreç süreleri, ilgili olasılık dağılımlarına bağlı olarak simüle edilmiştir. Yani, bir sürecin tamamlanma süresi, olasılıksal bir karakteristiğe sahip olduğundan ilgili olasılık değerlerine bağlıdır. Modelde, projenin bir sonrakinde hangi adımı atacağı yazı-tura ile belirlenmektedir. Dolayısıyla, o anda geliştirilen her bir proje parçası (aktif proje parçaları) için ayrı bir yazı-tura atılmalıdır.

Model, proje parçaları için önceden belirlenen *son teslim tarihleri*nin (deadline) aşılıp aşılmadığını kontrol etmektedir. Eğer son teslim tarihleri aşıldıysa, simülasyon koşusu durdurulur ve bu proje parçası için kalite değeri *sıfır* olarak kabul edilir. Dolayısıyla, bir simülasyonda birden fazla zaman aşımı meydana gelirse, projenin kalite seviyesi düşecektir. Zaman aşımı hem önceden belirlenen *son teslim tarihine* hem de bu proje sınıfına ait *olasılık dağılımı*na bağlıdır.

Simülasyon koşusu, ya projenin her bir parçası üzerindeki geliştirmelerin başarıyla tamamlanmasıyla ya da zaman aşımının meydana gelmesiyle sona erer.

*Simulasyon modeli*nin uygulaması için Arena®'nın ücretsiz olan akademik versiyonu kullanılmıştır. Fakat, varlık, varlık özellikleri ve sistem değişkenleri sayıları üzerindeki kısıtlamalardan dolayı, örnek proje için izlenmesi muhtemel her bir strateji için ayrı bir Arena® modeli geliştirilmiştir.

Modelin alternatif versiyonlarını karşılaştırmak ve onlar üzerinde bazı istatistiksel analizler yapmak için *Output Analyzer* aracı [20] kullanılmıştır.

4. Simülasyon Sonuçları ve Tartışmalar

Simülasyon sonuçlarını analiz etmek için şu çıktı ölçümleri kullanılmıştır: Ortalama kalite ödülü (average quality reward), ortalama geliştirme maliyeti (average development cost), ortalama geliştirme zamanı (average development time) ve ortalama takım kullanım oranı (average team utilization).

Modelde, proje geliştirirken izlenen *strateji* iki kısımdan oluşmaktadır: *proje parçalarının geliştirilme sırası* ve *takım atama stratejisi*. Her bir çıktı ölçüsü, her bir strateji için, %95 güven aralığında (confidence interval) alınmıştır. Tüm karşılaştırmalar, %95 güven aralığında Output Analyzer aracının *Paired-t test*i kullanılarak yapılmıştır. Bir başka deyişle karşılaştırmalar, 0.05 seviyesinde yapılmıştır. Fakat bazı karşılaştırmalar, daha derin analizler için 0.01 seviyesinde yapılmıştır (%99 güven aralığı). Karşılaştırmalar, *Wilcoxon test* veya *t-test* gibi diğer test metodları ve farklı güven aralıkları kullanılarak da yapılabilir.

Küçük bir örnek projede bile, yazılım geliştirme sürecinin olasılıksal doğası gereği, proje yöneticisinin hangi stratejiyi tercih etmesi gerektiği açık değildir. Özellikle, kalite seviyesi, geliştirme maliyeti ve geliştirme zamanı şansa bağlı olduğundan, görev tamamlama aktivitesine ilave belirsizlikler katılmaktadır.

Örnek proje dört parçaya sahip olduğundan, 24 (4 != 24) farklı *parça geliştirme sırası* vardır. Her bir parça geliştirme sırası için, çeşitli muhtemel takım atama stratejileri vardır. Örneğin, *ABCD parça geliştirme sırası* şunu ifade etmektedir: Öncelikle Takım 1, proje parçası A'yı ve Takım 2, proje parçası B'yi geliştirmek üzere atanmıştır. Bu iki takımdan işini erken bitiren takım (Takım 1 veya 2), proje parçası C'ye, diğeri de proje parçası D'ye atanacaktır. Dolayısıyla ABCD-1212 ve ABCD-1221 farklı stratejilerdir.

Toplamda, örnek projenin dört proje parçası ve iki proje takımı için mümkün olan 108 farklı takım atama stratejisi vardır. En iyi takım atama stratejisini bulmak için uygulama, 108 stratejinin her biri üzerinde 1000 defa koşulmuştur. Her bir koşu için kalite seviyesi, geliştirme maliyeti, geliştirme zamanı ve takım kullanım oranları gözlemlenmiştir.

4.1. Son Teslim Tarihinin Kalite Üzerine Etkisi

Son teslim tarihinin proje kalitesi üzerine etkisi, farklı iki senaryo kullanılarak incelenmiştir. Her bir senaryo için uygulama zamanı (execution time), bir son teslim tarihi ile sınırlandırılmıştır. Eğer bir simülasyon koşusu, son teslim tarihinden önce tamamlanamazsa, koşu durdurulur. 1^{nci} senaryo için son teslim tarihi, bazı simülasyon koşularının durdurulmasını garantileyecek şekilde yeterince küçük seçilmiştir. Fakat, 2^{nci} senaryo için son teslim tarihi, simülasyon koşularının tümünün tamamlanmasını garantileyecek şekilde yeterince büyük seçilmiştir.

Son teslim tarihinin proje (veya proje parçası) kalitesi üzerindeki etkisini analiz etmek için ABCD-1212 stratejisi takip edilmiştir. 1^{nci} ve 2^{nci} senaryoların sonuçları sırasıyla Tablo 6 ve Tablo 7'de gösterilmiştir.

Tablo 6: Son teslim tarihinin aşılması durumunda çıktı değerleri

Çıktı	Ortalama	Yarım Genişlik	Minimum Ortalama	Maksimum Ortalama
Parça A Ödül	93.58	1.06	25.00	100.00
Parça B Ödül	82.91	2.17	0.00	100.00
Parça C Ödül	67.37	2.77	0.00	100.00
Parça D Ödül	43.39	2.91	0.00	100.00
Parça A Maliyet	111.47	0.94	83.33	160.00
Parça B Maliyet	160.49	1.37	133.00	224.83
Parça C Maliyet	215.63	2.22	176.67	290.00
Parça D Maliyet	367.22	3.46	294.50	446.50
Toplam Maliyet	854.81	4.49	719.83	1069.50
Toplam Geliş-tirme Süresi	168.68	<1.18	137.00	206.04
Toplam Ödül	287.24	4.65	25.00	400.00

Tablo 6'de, proje parçaları B, C ve D için ödül değerinin minimum ortalamasının 0.00 olduğuna dikkat ediniz. Buradan şu sonuç çıkmaktadır: A parçası için son teslim tarihi hiç bir simülasyon koşusunda aşılmazken, B, C ve D parçaları için son teslim tarihi, bazı simülasyon koşularında aşılmıştır ve aşılan bu koşularda kalite değeri sıfır olarak kabul edilmiştir. Sonuç olarak, B, C ve D parçalarının *kalite ödül değerleri*nin ortalaması düşüktür ve simülasyon koşuları arasındaki değişikliklerin (variance) büyüklüğünü göstermek için %95 güven aralığındaki *yarım genişlik değerleri* (Half Width of the 95% Confidence Interval) yüksektir. Son teslim tarihinin, sadece kalite ödülünü etkilediğine ve geliştirme maliyeti üzerinde bir etkisi olmadığına dikkat ediniz.

Tablo 7'deki B, C ve D parçalarının *kalite ödül değerleri* Tablo 6'deki değerlerden daha yüksek; *yarım genişlik değerleri* ise koşular arasındaki değişikliklerin küçük olduğunu göstermek için daha düşüktür. Maliyet ve zaman değerleri her iki tabloda da aynıdır.

Tablo 6, ABCD-1212 stratejisi izlenerek geliştirilen projenin *kalite derecesi*ni 287.24/854.81 (birim maliyet başına elde edilen birim ödül miktarı) olarak göstermektedir. Tablo 7 ise, aynı strateji izlenerek geliştirlen aynı projenin kalite derecesini 348.65/854.81 olarak göstermektedir. 2^{nci} senaryodaki kalite derecesinin daha yüksek olduğu açıkça görülmektedir.

Tablo 7: Son teslim tarihinin aşılmaması durumunda çıktı değerleri

Çıktı	Ortalama	Yarım Genişlik	Minimum Ortalama	Maksimum Ortalama
Parça A Ödül	93.58	1.06	25.00	100.00
Parça B Ödül	92.41	1.13	25.00	100.00
Parça C Ödül	86.21	1.42	25.00	100.00
Parça D Ödül	76.46	1.64	25.00	100.00
Parça A Maliyet	111.47	0.94	83.33	160.00
Parça B Maliyet	160.49	1.37	133.00	224.83
Parça C Maliyet	215.63	2.22	176.67	290.00
Parça D Maliyet	367.22	3.46	294.50	446.50
Toplam Maliyet	854.81	4.49	719.83	1069.50
Toplam Geliş-tirme Süresi	168.68	<1.18	137.00	206.04
Toplam Ödül	348.65	2.59	175.00	400.00

4.2. En İyi ve En Kötü Stratejiler

Tablo 8, her bir çıktı ölçümleri için en iyi ve en kötü stratejileri ve gözlemlenen değerleri listelemektedir. Bu tabloya göre, en yüksek ortalama kalite değeri 400 üzerinden 349.97'dir. Bir proje parçası için en yüksek kalite değerinin 100 olduğuna ve dört parçaya sahip örnek proje için en yüksek toplam kalite değerinin 400 olacağına dikkat ediniz. Bu en yüksek ortalama kalite değeri, yalnız bir takım atama stratejisi (1221) ve dört geliştirme sırası (ABCD, ABDC, BACD ve BADC) tarafından karşılanmaktadır. Modelde, geliştirme esnasında tüm proje takımlarının meşgul olması gerektiğinden, Takım 1, A parçasını geliştirirken, Takım 2, proje parçası B'yi geliştirecektir. Bunun tersi de doğrudur. Dolayısıyla, geliştirme sırası AB'nin BA'dan bir farkı yoktur. Aynı durum, proje parçaları C ve D için de geçerlidir. Sonuç olarak, yukarıdaki dört geliştirme sırası, takım atama stratejisi 1221 için aynıdır.

En düşük ortalama kalite değeri 400 üzerinden 346.43'tür. Bu değer, yalnız bir takım atama stratejisi (1122) ve iki geliştirme sırası (ACBD ve CABD) tarafından karşılanmaktadır. Tüm proje takımları, her zaman meşgul olduğundan, bu takım atama stratejisi için geliştirme sıraları AC ve CA aynıdır.

Elde edilen en yüksek kalite değerinin, en düşük kalite değerinden istatistiksel olarak farklı olup olmadığını belirlemek için, ABCD-1221 ve ACBD-1122 stratejileri, *Output Analyzer* kullanılarak analiz edilmiştir. Sonuç, bu iki stratejinin 0.05 seviyesinde (%95 güven aralığında) birbirinden istatistiksel olarak farklı olduğunu göstermiştir. Fakat bu iki strateji, 0.01 seviyesinde (%99 güven aralığında) istatistiksel olarak birbirinden farklı değildir. Bu analiz için *Paired-t Test* kullanılmıştır. Çünkü Paired-t yaklaşımı, özellikle, senaryolar istatistiksel olarak birbirinden bağımsız değilse ve hassasiyet derecesi çok önemli değilse, daha çok kullanılmaktadır [20]. En yüksek kalite değerinin, en düşük kalite değerinden istatistiksel olarak farklı olmamasının sebebi, başarısız olma oranı (fail rate) ile başarılı olma oranının (success rate) olasılık tablolarında Takım 1 ve Takım 2 için aynı tanımlanmış olmasıdır. Bundan dolayı, Takım 1 tarafından geliştirilen bir parçanın kalite seviyesinin, aynı parçanın Takım 2 ile geliştirilmesi durumundaki kalite seviyesiyle yaklaşık aynı olması beklenmektedir. Tablo 8'teki stratejilerin kalite seviyeleri arasındaki fark, simulasyon modelinde kullanılan rasgele (random) girdilerin bir sonucudur. Sonuç olarak, tüm stratejilerin kalite seviyeleri, birbirlerine çok yakın

olduğundan, en iyi ve en kötü stratejiler seçilirken kalite değeri dikkate alınmayacaktır.

En yüksek ortalama geliştirme maliyeti 893.15 birimdir. Geliştirme maliyeti, sadece geliştirme süresine bağlı değildir; aynı zamanda geliştirmeyi yapan takımların birim maliyet değerlerine de bağlıdır. En yüksek geliştirme maliyeti, yalnız bir takım atama stratejisi (2212) ve iki geliştirme sırası (BCAD ve CBAD) tarafından karşılanmaktadır. Bu iki geliştirme sırasının, takım atama stratejisi 2212 için aynı olduğuna dikkat ediniz. Her iki geliştirme sırasında da, Takım 2 ilk önce B parçasını (Takım 1, sadece C parçasını geliştirirken), sonra A'yı ve son olarak da D'yi geliştirmektedir. Bir başka deyişle, projenin iki küçük parçası ve en büyük parçası Takım 2 tarafından geliştirilmektedir. Kullanılan olasılık dağılımlarına göre, Takım 2, Takım 1'den daha yavaş ve daha ucuzdur. Bunu akılda tutarak şu sonuca varılabilir: Örnek projede, yavaş takımın projenin üç parçasını geliştirdiği stratejiler, yüksek ortalama geliştirme maliyetine sahip stratejilerdir.

En düşük ortalama geliştirme maliyeti 771.31 birimdir. Bu değer, yalnız bir takım atama stratejisi (1121) ve iki geliştirme sırası (CDAB ve DCAB) tarafından karşılanmaktadır. Bu iki geliştirme sırasının, takım atama stratejisi 1121 için aynı olduğuna dikkat ediniz. Her iki geliştirme sırasında da, Takım 1 önce D parçasını (Takım 2, sadece C parçasını geliştirirken), sonra A'yı ve son olarak da B'yi geliştirmektedir. Bir başka deyişle, projenin iki küçük parçası ve en büyük parçası Takım 1 tarafından geliştirilmektedir. Buradan şu sonuca varılabilir: Örnek projede, hızlı takımın projenin üç parçasını geliştirdiği stratejiler, düşük ortalama geliştirme maliyetine sahip stratejilerdir.

BCAD-2212 ve CDAB-1121 stratejileri, *Paired-t Test* kullanılarak Output Analyzer ile incelenmiştir. Sonuç, bu iki stratejinin 0.05 seviyesinde istatistiksel olarak birbirinden farklı olduğunu göstermiştir.

Tablo 8: En iyi ve en kötü stratejiler

Çıktı	Değer	Takım Atama Stratejisi	Parça Geliştirme Sırası
En Yüksek Ortalama Kalite Ödülü	349.97	1221	ABCD ABDC BACD BADC
En Düşük Ortalama Kalite Ödülü	346.43	1122	ACBD CABD
En Yüksek Ortalama Geliştirme Maliyeti	893.15	2212	BCAD CBAD
En Düşük Ortalama Geliştirme Maliyeti	771.31	1121	DCAB CDAB
En Uzun Ortalama Geliştirme Süresi (gün)	213.73	2212	BCAD CBAD
En Kısa Ortalama Geliştirme Süresi (gün)	131.79	1112	ADBC BDAC DABC DBAC
En Düşük Ortalama Takım Kullanım Oranları Arasındaki Fark	11.10%	1112	CDAB DCAB
En Kötü Ortalama Takım Kullanım Oranları Arasındaki Fark	69.33%	2212	BCAD CBAD

En uzun ortalama geliştirme süresi 213.73 gündür. Geliştirme süresi, proje takımlarının olasılık dağılımlarına bağlıdır. En uzun geliştirme süresi, yalnız bir takım atama stratejisi (2212) ve iki geliştirme sırası (BCAD ve CBAD) tarafından karşılanmıştır. Bu iki geliştirme sırasının, takım atama stratejisi 2212 için aynı olduğuna dikkat ediniz. Takım atama stratejisi 2212, iki küçük parçanın ve en büyük parçanın Takım 2 tarafından geliştirildiği anlamına gelmektedir. Bundan dolayı şu sonuca varılabir: Örnek projede, yavaş takımın projenin üç parçasını geliştirdiği stratejiler, uzun ortalama geliştirme süresine sahip stratejilerdir.

En kısa ortalama geliştirme süresi 131.79 gündür. Bu değer, yalnız bir takım atama stratejisi (1112) ve dört geliştirme sırası tarafından karşılanmaktadır. Bu takım atama stratejisinde, yukarıdaki dört geliştirme sırası için gözlemlenen tüm değerler birbirlerinden farklı ama birbirlerine çok yakındır. Bu değerlerin istatistiksel olarak birbirlerinden farklı olup olmadıklarını anlamak için Output Analyzer kullanılmıştır. Sonuç, bu değerlerin 0.05 seviyesinde birbirlerinden istatistiksel olarak farklı olmadıklarını göstermiştir. Bu stratejilerde, örnek projenin iki küçük parçası ile büyük parçalarından biri Takım 1 tarafından geliştirilmiştir. Buradan şu sonuca varılabilir: Örnek projede hızlı takımın projenin üç parçasını geliştirdiği stratejiler, kısa ortalama geliştirme süresine sahip stratejilerdir.

BCAD-2212 ve ADBC-1112 stratejileri, *Paired-t Test* ile Output Analyzer kullanılarak incelenmiştir. Sonuç, bu iki stratejinin 0.05 seviyesinde birbirlerinden istatistiksel olarak farklı olduklarını göstermiştir.

4.3. En İyi 3 Strateji

En iyi strateji, en yüksek ortalama kalite değerine, en düşük ortalama geliştirme maliyetine, en kısa ortalama geliştirme süresine ve en iyi takım kullanım oranına sahip olmalıdır.

Tablo 8'de listelenen hiç bir strateji bu dört şartı aynı anda sağlamamaktadır. Tablo 9, Tablo 10 ve Tablo 11 sırasıyla, geliştirme maliyeti, geliştirme süresi ve takım kullanım oranı açısından en iyi üç takım atama stratejisini listelemektedir.

En iyi strateji, aşağıdaki üç tabloda da 1^{nci} sırada olmalıdır. Fakat hiç bir strateji bu şartı sağlamamaktadır. Bundan dolayı en iyi stratejiyi seçmek için iki farklı yaklaşım izlenebilir.

Tablo 9: Geliştime maliyeti açısından en iyi 3 takım atama stratejisi

Takım Atama Stratejisi	Geliştirme Sırası	En Düşük Geliştirme Maliyeti
1121	CDAB, DCAB	771.31
	DCBA, CDBA	771.31
	BCAD, CBAD	775.84
	ACBD, CABD	776.28
	BCDA, CBDA	778.26
	ACDB, CADB	778.38
2211	BCAD, CBAD	801.98
	BCDA, CBDA	801.98
	ACBD, CABD	802.58
	ACDB, CADB	802.58
	ADBC, DABC	803.45
	ADCB, DACB	803.45
	BDAC, DBAC	803.52
	BDCA, DBCA	803.52
1112	BDAC, DBAC	807.41
	ADBC, DABC	807.43
	ADCB, DACB	808.92
	BDCA, DBCA	809.67
	CDAB, DCAB	810.29
	CDBA, DCBA	810.99

Tablo 10: Geliştime süresi açısından en iyi 3 takım atama stratejisi

Takım Atama Stratejisi	Geliştirme Sırası	En Kısa Geliştirme Süresi (gün)
1112	BDAC, DBAC	131.79
	ADBC, DABC	131.79
	CDAB, DCAB	132.12
	CDBA, DCBA	132.25
	BDCA, DBCA	132.36
	ADCB, DACB	132.43
2121	ADBC, DABC	134.15
	ADCB, DACB	134.15
	ABCD, BACD	134.19
	ABDC, BADC	134.19
	CDAB, DCAB	134.88
	CDBA, DCBA	134.88
	BCDA, CBDA	134.97
1221	BDCA, DBCA	136.30
	BDAC, DBAC	136.30
	ABCD, BACD	136.72
	ABDC, BADC	136.72
	CDAB, DCAB	136.99
	CDBA, DCBA	136.99
	ACDB, CADB	137.02

Tablo 11: Takım kullanım oranı açısından en iyi 3 takım atama stratejisi

Takım Atama Stratejisi	Geliştirme Sırası	En İyi Takım Kullanım Oranları Farkı (%)
1112	CDAB, DCAB	11.10
	CDBA, DCBA	11.12
	BDAC, DBAC	11.26
	ADBC, DABC	11.27
	BDCA, DBCA	11.59
	ADCB, DACB	11.86
2121	ADBC, DABC	12.20
	ADCB, DACB	12.20
	ABCD, BACD	12.39
	ABDC, BADC	12.39
	BCDA, CBDA	12.80
	CDAB, DCAB	13.42
	CDBA, DCBA	13.42
1221	BDCA, DBCA	13.67
	BDAC, DBAC	13.67
	ABCD, BACD	14.00
	ABDC, BADC	14.00
	ACDB, CADB	14.05
	CDAB, DCAB	14.75
	CDBA, DCBA	14.75

İlk yaklaşım, geliştirme maliyeti üzerine kurulmuştur. Proje geliştirme üzerinde, geliştirme süresinin önemli bir etkisi vardır ve bu durum, tüm proje parçaları için bir son teslim tarihi belirlenerek ve geliştirme maliyeti süreye bağlı bir şekilde hesaplanarak dikkate alınmıştır. Eğer son teslim tarihi aşılırsa, projenin kalite seviyesi düşük olacaktır. Dolayısıyla, en yüksek kalite seviyesi seçilirse, proje tamamlandığında en azından son teslim tarihinin aşılma ihtimali en düşük olacaktır. Son teslim tarihinin aşılmadığı durumlarda, geliştirme maliyeti, geliştirme süresinden daha önemlidir. Dolayısıyla strateji, en düşük geliştirme maliyetine sahip olacak şekilde seçilmelidir. Sonuç olarak, izlenen stratejilerin kalite seviyeleri arasında önemli bir fark olmadığından, en iyi takım atama stratejisi olarak 1121 seçilebilir. Modeldeki rasgelelik (randomness) yüzünden, sadece bir tek stratejinin en iyi olarak belirlenmesi tam olarak doğru olmaz. Bundan dolayı, sadece bir tek strateji seçmek yerine en az üç tane strateji belirlemek daha doğru olacaktır. Buna göre, en iyi ilk üç strateji şöyle listelenebilir: CDAB-1121 (DCAB-1121), CDBA-1121 (DCBA-1121) ve BCAD-1121 (CBAD-1121). Bu stratejiler için takım kullanım oranları en iyisi değildir; fakat, birim maliyet (unit cost) başına elde edilen kalite değeri (quality value), yani kalite derecesi (quality degree), en önemli faktör olduğundan, takım kullanım oranlarının en iyisi olması gerekmemektedir. Örneğin, strateji CDAB-1121 en yüksek kalite derecesine sahiptir: 349.84/771.31.

2nci yaklaşım geliştirme maliyeti, geliştirme süresi ve takım kullanım oranı üzerine kurulmuştur. Bu üç unsur dikkate alındığında, en uygun strateji bu üç unsurun hepsini karşılayacak şekilde seçilmelidir. Sadece bir tek takım atama stratejisi, yukarıdaki üç tabloda da aynı anda mevcuttur: 1112. Bu strateji için ortalama geliştirme süresi 131.79 gün, ortalama geliştirme maliyeti 807.41 birim ve ortalama takım kullanım oranı Takım 1 için %97.69, Takım 2 için %86.10'dur (takım kullanım oranları arasındaki fark %11.26'dır). Bu yaklaşıma göre, en iyi üç strateji BDAC-1112 (DBAC-1112), ADBC-1112 (DABC-1112) ve CDAB-1112 (DCAB-1112) olarak seçilebilir.

Eğer takımların kullanım oranları birbirine yakınsa, yani takım kullanım oranları arasındaki fark küçük ise, takım ataması dengeli (balanced) yapılmış demektir; çünkü takımların yaptıkları işlerin büyüklüğü, takımların yetenek seviyeleri ile dengelenmiştir. Bu, projelerde takımları barış ve huzur içinde tutmak için gerekli diğer bir etkendir. Dolayısıyla projelerde, dengeli takım atamaları tercih edilmelidir. Önerilen modelin, dengeli takım atamalarında da yöneticilere yol gösterici olduğu aşikardır.

Eğer en kısa geliştirme süresi en büyük öneme sahipse, 2nci yaklaşım izlenebilir. Fakat, son teslim tarihi aşılmıyorsa ve bundan dolayı geliştirme süresinin en kısa olması gerekmiyorsa (yani, zaman üzerindeki tek kısıtlama, son teslim tarihinin aşılmaması ise) 1nci yaklaşım izlenebilir.

Her iki yaklaşım da şunu göstermektedir: Örnek projedeki her bir takımın, iki proje parçasını geliştirmesini sağlamak (ilk bakışta bu strateji, en iyisi olarak görülebilmektedir) yerine, yetenek seviyesi yüksek olan takımı projenin iki küçük parçasına ve büyük parçalarından birine atamak ve yetenek seviyesi düşük olan takımı, projenin diğer büyük parçasına atamak daha iyi sonuçlar vermektedir. Bkz. Tablo 9, 10 ve 11.

Tablo 10 ve 11'e göre takım kullanım oranları arasındaki fark ne kadar düşükse, geliştirme süresi o kadar kısadır. Örneğin, takım atama stratejisi 1112, hem en iyi takım kullanım oranına hem de en kısa geliştirme süresine sahiptir. Fakat benzer ilişki Tablo 9 ile Tablo 10 arasında yoktur. Çünkü, geliştirme maliyeti sadece geliştirme süresine bağlı değildir; aynı zamanda takımların birim maliyet değerlerine de bağlıdır.

4.4. En Kötü 3 Strateji

En kötü strateji, en düşük ortalama kalite değerine, en yüksek ortalama geliştirme maliyetine, en uzun ortalama geliştirme süresine ve en kötü ortalama takım kullanım oranına sahip olmalıdır.

Tablo 8'deki hiç bir strateji bu dört unsuru aynı anda sağlamamaktadır. Tablo 12, Tablo 13 ve Tablo 14 sırasıyla, geliştirme maliyeti, geliştirme süresi ve takım kullanım oranı açısından en kötü üç takım atama stratejisini listelemektedir.

En kötü takım atama stratejisi, bu üç tablonun da 1nci stratejisi olmalıdır: 2212. Bu takım atama stratejisi, geliştirme sıraları BCAD, CBAD, ACBD ve CABD'ye uygulanabilir. Dolayısıyla en kötü üç strateji BCAD-2212 (CBAD-2212), ACBD-2212 (CABD-2212) ve ACBD-1122 (CABD-1122) olarak seçilebilir. En kötü üç stratejiyi belirlerken izlenen yaklaşımın, en iyi üç stratejiyi belirlerken izlenen 2nci yaklaşım ile aynı olduğuna dikkat ediniz.

Tablo 12, Tablo 13 ve Tablo 14'te, ADBC-1122 stratejisi ile DABC-1122 stratejisi aynı çıktı değerlerine sahip değildir; bunun nedeni Arena® aracının rasgele girdi üreticisinin (random input generator), her kullanımda farklı rasgele girdi kümeleri (random input sets) kullanmasıdır (bkz. Bölüm 12 [20]). ADBC-DABC strateji çifti, 0.05 seviyesinde istatistiksel olarak farklıdır ama 0.01 seviyesinde farklı değildir.

Sonuç olarak; yetenek seviyesi düşük olan takımın, projenin iki büyük parçasıyla küçük parçalarından birine atanması, geliştirme maliyeti, geliştirme süresi ve takım kullanım oranları açısından kötü bir stratejidir. Diğer kötü stratejiler ise yetenek seviyesi düşük olan takımın projenin iki büyük parçasına veya projenin iki küçük parçasına ve büyük parçalarından birine atanmasıdır.

Tablo 12: Geliştime maliyeti açısından en kötü 3 takım atama stratejisi

Takım Atama Stratejisi	Geliştirme Sırası	En Yüksek Geliştirme Maliyeti
2212	BCAD, CBAD	893.15
	ACBD, CABD	893.14
1122	ACBD, CABD	867.64
	BCAD, CBAD	867.40
	DABC	865.79
	BDAC, DBAC	865.36
	ADBC	865.23
2221	BDAC, DBAC	861.62
	ADBC, DABC	861.55

Tablo 13: Geliştime süresi açısından en kötü 3 takım atama stratejisi

Takım Atama Stratejisi	Geliştirme Sırası	En Uzun Geliştirme Süresi (gün)
2212	BCAD, CBAD	213.73
	ACBD, CABD	213.60
1122	ACBD, CABD	202.91
	BCAD, CBAD	202.91
	DABC	202.34
	BDAC, DBAC	202.34
	ADBC	202.17
2221	BDAC, DBAC	184.71
	ADBC, DABC	184.58

Tablo 14: Takım kullanım oranı açısından en kötü 3 takım atama stratejisi

Takım Atama Stratejisi	Geliştirme Sırası	En Kötü Takım Kullanım Oranları Farkı (%)
2212	BCAD, CBAD	69.33
	ACBD, CABD	69.26
1122	BCAD, CBAD	66.36
	ACBD, CABD	66.32
	BDAC,DBAC	66.30
	DABC	66.23
	ADBC	66.20
2221	BDAC, DBAC	54.73
	ADBC, DABC	54.65

4.5. Takım Atama Stratejilerinin Performansı

Yetenek seviyesi yüksek takımın, projenin iki küçük parçasıyla büyük parçalarından birini geliştirdiği ve yetenek seviyesi düşük olan takımın, projenin diğer büyük parçasını geliştirdiği takım atama stratejileri, en fazla tercih edilmesi gereken stratejilerdir. Örneğin, 1112 ve 1121. Eğer yetenek seviyesi düşük takım, en büyük proje parçası (proje parçası D) üzerinde çalışıyorsa en kısa geliştirme süresinin elde edildiğine dikkat ediniz. Dolayısıyla strateji 1112, strateji 1121'e göre daha fazla tercih edilir durumdadır.

Yetenek seviyesi düşük takımın, projenin iki küçük parçasıyla büyük parçalarından birini geliştirdiği ve yetenek seviyesi yüksek olan takımın, projenin diğer büyük parçasını geliştirdiği takım atama stratejileri en az tercih edilmesi gereken stratejilerdir. Örneğin, 2212 ve 2221. Eğer yetenek seviyesi düşük takım, en büyük proje parçası (proje parçası D) ile projenin iki küçük parçasını geliştiriyorsa en uzun geliştirme süresinin elde edildiğine dikkat ediniz. Dolayısıyla strateji 2212, strateji 2221'e göre daha az tercih edilir durumdadır.

Yetenek seviyesi yüksek takımın, projenin küçük parçalarını, yetenek seviyesi düşük takımın ise projenin

büyük parçalarını geliştirdiği stratejiler de az tercih edilmesi gereken stratejiler arasındadır. Örnegin, strateji 1122.

Yetenek seviyesi düşük takımın, projenin küçük parçalarını, yetenek seviyesi yüksek takımın ise projenin büyük parçalarını geliştirdiği stratejiler, bir önceki strateji grubuna göre daha fazla tercih edilebilir. Örneğin, strateji 2211, strateji 1122'ye göre daha fazla tercih edilebilir durumdadır.

Her bir takımın, projenin bir büyük bir de küçük parçasını geliştirdiği, fakat yetenek seviyesi yüksek takımın en büyük parçayı geliştirdiği stratejiler (örneğin, strateji 1221 ve 2121), yetenek seviyesi düşük takımın en büyük parçayı geliştirdiği stratejilerden (örneğin, 1212 ve 2112) daha iyi bir performansa sahiptir. Strateji 2121'in, strateji 1221'den daha iyi bir performasa sahip olduğuna dikkat ediniz. Benzer şekilde, strateji 2112, strateji 1212'den daha iyi bir performansa sahiptir. Bunun tek istisnası CDBA (DCBA) geliştirme sırası takip edildiğinde görülmektedir. Fakat, bu istisnai durumdaki performans farkı istatistiksel olarak önemli değildir.

4.6. Geliştirme Sıralarının Performansı

Ortalama geliştirme sürelerine göre, geliştirme sıralarının performası Tablo 15'te listelenmiştir. Bu tabloya göre en kısa geliştirme süresi 131.79 gündür ve ADBC, BDAC, DABC ve DBAC geliştirme sıralarına aittir. En büyük proje parçası (D) ve projenin küçük parçalarından biri (A veya B) ilk önce geliştirildiğinde, geliştirme süresinin en kısa olduğuna dikkat ediniz.

Tablo 15: Geliştirme sıralarının performası

Geliştirme Sırası	En Kısa Ortalama Geliştirme Süresi (gün)	En Uzun Ortalama Geliştirme Süresi (gün)
ABCD	134.19	166.68
ABDC	134.19	166.68
ACBD	147.57	213.60
ACDB	137.02	163.37
ADBC	131.79	202.17
ADCB	132.43	167.27
BACD	134.19	166.68
BADC	134.19	166.68
BCAD	147.41	213.73
BCDA	134.97	165.62
BDAC	131.79	202.34
BDCA	132.36	165.44
CABD	147.57	213.60
CADB	137.02	163.37
CBAD	147.41	213.73
CBDA	134.97	165.62
CDAB	132.12	167.75
CDBA	132.25	165.92
DABC	131.79	202.34
DACB	132.43	167.27
DBAC	131.79	202.34
DBCA	132.36	165.44
DCAB	132.12	167.75
DCBA	132.25	165.92

En uzun geliştirme süresi 213.73 gündür ve BCAD ve CBAD geliştirme sıralarına aittir. Projenin 2^{nci} en büyük parçası (C) ile projenin küçük parçalarından biri (A veya B) ilk önce geliştirildiğinde, geliştirme süresinin en uzun olduğuna dikkat ediniz.

Fakat bu analiz, takım atama stratejilerini hesaba katmadan tam olamaz. Çünkü, en yavaş geliştirme sırası (BCAD, CBAD), en hızlı geliştirme sırasından (ADBC, BDAC, DABC, DBAC) daha hızlı olabilmektedir. Örneğin, en hızlı geliştirme sıralarından biri olan ADBC, takım atama stratejisi 1122 ile 202.17 gün iken; en yavaş geliştirme sıralarından biri olan BCAD, takım atama stratejisi 2211 ile 147.41 gündür.

Benzer sonuçlar diğer çıktı ölçümlerinden de türetilebilir.

5. Sonuç ve Öneriler

Bu çalışmada, yazılım geliştirme sürecini yazılım kalitesi bakış açısıyla modellemek için daha önce önerilen MDP tabanlı matematiksel modelin [16] bir *simülasyon modeli* olarak uygulanmasının detayları açıklanmış, simülasyon modeli örnek bir proje üzerinde çalıştırılmış ve simülasyonun sonuçları detaylarıyla incelenmiştir.

Model, bir *ayrık zamanlı, sonlu ufuklu Markov Karar Süreci* (discrete-time finite-horizon Markov Decision Process) modelidir [16]. Çünkü model, *Markov özelliklerine* (Markov property) sahip *stokastik* bir süreci temsil etmektedir. Yazılım geliştirme süreci için stokastik model, deterministik modele göre daha uygundur; çünkü deterministik modelde herşey en baştan kesin olduğundan, görev sürelerinin ve meydana gelen olayların belirsizliği gibi durumlar hesaba katılamamaktadır.

Model, zaman ekseni üzerinde fazları, sistemin *durum*larını (states), sistemdeki *eylem*leri (actions), geçiş olasılıklarını, geçiş maliyetlerini (cost) ve geçiş ödüllerini (rewards) içermektedir [16].

Model, projedeki görev atamalarını, takımların ve YKG unsurlarının yetenek seviyelerini, ve hataların sebep olduğu *yeniden çalışma* (rework) işlerini temsil etmektedir. Yazılım geliştirme fazlarının süresindeki ve fazlar boyunca olayların meydana gelmesindeki belirsizliği hesaba katmak için, modelin *stokastik* doğası *temel olasılıklar* ile modellenmiştir. *Temel olasılıklar*, proje takımlarının ve YKG unsurlarının yetenek seviyelerini temsil etmektedir. Model, yazılım geliştirmede kullanılabilecek farklı politikaları denemeyi kolaylaştırdığından, projelerinin kalitesini, geliştirmenin erken safhalarında değerlendirmek için kullanılabilir. Yönetici, izlenecek stratejiyi değiştirdiği zaman, kalite seviyesinin nasıl değiştiğini çabucak görebildiği için, farklı stratejileri karşılaştırarak değerlendirebilir ve kalite açısından en iyi stratejiyi seçebilir.

Uygulama, yazılım geliştirme sürecinin sadece tasarım fazını içerdiğinden, önerilen modelin basitleştirilmiş bir halidir. Dolayısıyla, YKG sistemini oluşturma stratejisi uygulama için bir girdi değildir.

Modelinin uygulaması, Arena® simülasyon aracı kullanılarak yapılmıştır. Yazılım projelerini geliştirmede izlenecek çeşitli stratejilerin performaslarının, yazılım kalitesi açısından nasıl analiz edilebileceğini göstermek için, uygulama literatürden seçilen basit bir örnek proje [6] üzerinde çalıştırılmıştır.

Bir proje parçası geliştirilirken izlenen yol ile elde edilen geliştirme süresi ve maliyeti, olasılık değerlerine ve olasılık dağılımlarına bağlıdır. Olasılık değerlerinde ve dağılımlarında, proje parçaları ve takımlar dikkate alınmıştır. Bir proje parçası için toplam süre/maliyet, parçanın geliştirilme süresinin/maliyetinin, KG aktivitesinin uygulanma süresinin/maliyetinin ve yeniden çalışma süresinin/maliyetinin toplamıdır. *Toplam kalite ödülü* (total quality reward) temel olasılıklara bağlı iken, *net kalite*

derecesi (net quality degree), toplam maliyet ile toplam kalite ödülü dikkate alınarak belirlenir. Dolayısıyla, aynı kalite ödülüne sahip iki stratejiden, geliştirme maliyeti düşük olan strateji tercih edilmelidir.

Genel olarak, bu makalede sunulan durum çalışması, önerilen modelin, verilen varsayımlar altında, en yüksek kalite seviyesine ulaşmak için, en iyi takım atama stratejisini belirleme kabiliyetine sahip olduğunu göstermiştir. Bu çalışma, daha başka stratejiler içerecek şekilde genişletilebilir. Bu model, orta veya üst olgunluk derecesine sahip yazılım geliştirme organizasyonlarında daha verimli bir şekilde kullanılabilir; çünkü bu organizasyonlar normalde, kendilerine özgü geliştirme ortamlarını yansıtan daha önce geliştirilmiş projelerden toplanmış gerçek datalara sahiptir.

Bu çalışmada sunulan MDP tabanlı simülasyon modeli, yazılım geliştirme sürecinin sadece tasarım fazını içermektedir. Model, aynı yol izlenerek tam bir sistem modeli elde etmek için, yazılım geliştirme sürecinin tüm fazlarını içerecek şekilde genişletilebilir. Model, ayrıca varsayımlardan bir veya birkaçında iyiliştirme yapılarak da geliştirilebilir. Örneğin, model üzerinde aşağıdaki iyileştirmelerden biri veya birkaçı yapılabilir:

1. Modelin, *Ardışık ve Artan Geliştirme Süreci* (Iterative & Incremental Development Process) için uyarlanması,
2. Parçaların birbirlerine etkisi hesaba katılması,
3. Niteleme faaliyetini gerçekleştiren takımın yetenek seviyesinin, niteleme faaliyetinin sonucunu etkileyebilmesi,
4. Bir proje parçasına birden çok takımın atanabilmesi,
5. Bir proje bölümünün her bir elemanı hakkında istatistiksel veriler toplanarak, her bir eleman için temel olasılıkların hesaplanması (yani, takım yerine ferdi hesaplamaların yapılması),
6. Yazılım kalitesi, geliştirme süresi/maliyeti üzerinde muhtemel proje risklerinin etkisini incelemek için bir risk modelinin eklenmesi.

6. Kaynakça

[1] Pressman, Roger S. (2000) *Software Engineering - A Practitioner's Approach*, European Adaptation by D. Ince, 5th Edition, McGraw Hill International, London.

[2] Galin, Daniel (2004) *Software Quality Assurance – From theory to implementation*, Pearson – Addison Wesley, England.

[3] Ambler, Scott W. and Jeffries, Ron (2001) *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*, 1st Edition, John Wiley & Sons.

[4] Boehm, B. W. (1981) *Software Engineering Economics*, Prentice-Hall, NJ, USA.

[5] Balan, S. *A Composite Model for Software Quality Assurance,* from the book *Software Engineering-Concepts and Applications* by Sheshasaayee, J.G. The ICFAI University Press, 2006, Hyderabad, India

[6] Padberg, Frank (2002) *A Discrete Simulation Model for Assessing Software Project Scheduling Policies,* Software Process Improvement and Practice 7:127-139, John Wiley Sons, Ltd.

[7] Padberg, Frank (2000) *Estimating the Impact of the Programming Language on the Development Time of a Software Project*, Proceedings International Software Development and Management Conference ISDM/AP-SEPG 287-298.

[8] Padberg, Frank (2000) *Towards Optimizing the Schedule of Software Projects with Respect to Development Time and Cost*, International Software Process Simulation Modeling Workshop ProSim.

[9] Drabick, Rodger (2000), *A Process Model for Software Quality Assurance/Software Quality Engineering,* Volume 2, Number 4, American Society for Quality (ASQ).

[10] Prabhakar, Arati (2003), *Establishing and Implementing Software Quality Assurance,* Institute of Internet Quality, Virginia, USA.

[11] Tamres, Louise (2004), *Quick Start to Quality – Five Important Test Support Practices*, International Conference on Software Testing Analysis & Review (STAR*EAST*) by Software Quality Engineering, May 17-21, 2004, Orlando, FL, USA.

[12] Ntourntoufis, Panos (2002), *From Software Quality Control to Quality Assurance*,UPSPRING Software, UK.

[13] Harris, Margaret (2003), *Quality Assurance Section for a Design Specification*, http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=ART&ObjectId=6552 (Last visited date: 16 March 2007)

[14] Srivastava, Ambrish K.; Sharma, Gopesh; Keeni, Gargi (2002), *A Systematic Approach to Plan Inspection Process*, The 7th International Symposium on Future Technology (ISFST), October 20-25, 2002 Xi'an, Wuhan, China.

[15] Hall T. et. al. (2002), *Implementing Software Process Improvement: An Empirical Study*, in: Software Process Improvement and Practice, 2002, 7, pp. 3-15.

[16] Korkmaz, Ömer; Akman, İbrahim (2007), *Yazılım Kalitesini Değerlendirmek İçin Bir Simülasyon Modeli*, s. 125-134, 3üncü UYMS, 27-30 Eylül 2007, Bilkent Üniversitesi, Ankara.

[17] ARENA® Rockwell Automation (2007), *Forward Visibility For Your Business*, http://www.arenasimulation.com/ (Last visited date: 12 July 2007).

[18] ARENA® Rockwell Automation (2007), *Forward Visibility For Your Business*, http://www.arenasimulation.com/ (Last visited date: 12 July 2007).

[19] ARENA® Rockwell Automation (2007), *Product Overview*, http://www.arenasimulation.com/products/default.asp (Last visited date: 01 July 2007).

[20] Kelton, W. D.; Sadowski, R. P.; Sturrock, D. T. (2007), *Simulation with Arena*, 4th Edition, McGraw Hill International, ISBN-13:978-0-07-110685-6.