UML Diyagramları Kullanımının Yazılım Gerekleri Gözden Geçirme Performansına Etkileri: Bir Replikasyon Çalışması

Özlem Albayrak

Bilgisayar Teknolojisi ve Bilişim Sistemleri
İhsan Doğramacı Bilkent Üniversitesi
Ankara, Türkiye

ozlemal@bilkent.edu.tr

Özet. Bu çalışmada yazılım gerekleri spesifikasyonu dokümanına dahil edilen UML diyagramlarının, yazılım gerekleri gözden geçirme sürecinde raporlanan, ve doğru olarak belirlenen hata sayılarına etkilerini inceleyen kontrollü deney sonuçları sunulmaktadır. Bir replikasyon çalışması olan bu çalışmanın amacı, konuyla ilgili daha önce yapılmış özgün çalışma sonuçlarının geçerliliğini artırmaktır. Replikasyonda özgün çalışmada kullanılanlardan farklı materyaller kullanılmıştır. Çalışmaya katılan 66 son sınıf lisans öğrencisinin tamamı uygulamalı yazılım mühendisliği dersi ve bu dersin önkoşulu olan yazılım mühendisliği derslerini zorunlu olarak almıştır.

Çalışma bulgularına göre, yazılım gerekleri belirtim dokümanında UML kullanım durumları ve UML sınıf diyagramlarının bulunması katılımcıların yazılım gerekleri gözden geçirme sürecinde raporladıkları hata sayısını anlamlı olarak etkilemektedir. Yazılım gereklerini gözden geçirenler UML diyagramların dahil edilmesi durumunda daha fazla sayıda doğru hata belirlemektedir. UML diyagramlarının dahil edilmesiyle elde edilen olumlu etki, akademik başarı notları düşük olan öğrencilerde daha fazla olarak gözlemlenmiştir. Deneyde kullanılan gereksinim dokümanlarının hangi sıra ile kullanıldığının raporlanan hata sayısı üstünde anlamlı bir etkisi vardır. UML diyamların kullanılması ile hata bulmak için harcanan zaman arasında da anlamlı bir ilişki gözlemlenmiştir. Hem özgün, hem de replikasyon çalışması UML diyagramlarının dahil edilmesi durumunda daha fazla sayıda hata raporlandığını göstermektedir.

Anahtar Kelimeler: yazılım gözden geçirme; UML; gereksinim gözden geçirme

1 Giriş

IEEE Standard for Reviews and Audits 1028:2008'e göre gözden geçirme "bir yazılım ürününün hata bulma ve yazılımla ilgili standart ve spesifikasyonlardan sapma da dahil olmak üzere anormallikleri belirleme amacıyla görsel olarak incelenmesidir [1]. Yazılım gerekleri ile ilgili hataların belirlenmesi, yazılım mühendisliğinde yazılım kalite güvencesi teknikleri içinde en etkin yöntemlerdendir [2, 3].

Unified Modeling Language (UML) diyagramları yazılım geliştirmede sıklıkla kullanılır [5, 6] ve görsellik ve modelleme desteği verir [4]. Nesne modelleri analistler ve kullanıcılar arasındaki iletişim problemlerini en aza indirger [5]. Pratikte, bilişim sistemleri gereksinim dokümanlarında UML diyagramlarına rastlamak yaygındır [7].

Gereksinimlerdeki hatalar ilerleyen aşamaları da olumsuz etkileyeceğinden maliyeti yüksek hatalardır [8, 9]. Bu hatalar zamanında belirlenmezse, ileride çok daha yüksek maliyetlere neden olacaklardır. Yazılım gözden geçirme alanında yapılan bilimsel çalışmaların birincil amacı süreçlerin etkenlik ve etkinliklerinin artırılmasıdır [9]. UML ve gözden geçirme ile ilgili olarak daha önce yapılan araştırmalar UML diyagramları bulunan dokümanların gözden geçirilmesi sürecinin etkinleştirilmesine dönük yolların bulunmasında odaklanmıştır. Literatürde gözden geçirme sürecinde farklı okuma tekniklerinin hangilerinin daha etkin olduğunu inceleyen çok sayıda araştırma yapılmıştır [2, 8, 10, 11, 12]. Aynı okuma tekniği kullanıldığında bile bireysel etkinlikler arasında farklılıklar gözlemlenmiştir [10].

Bu çalışmanın asıl odağı gözden geçirme sürecinde kullanılan dokümanlardır. Çalışmada okuma tekniği ve gözden geçirenleri sabit tutulmuş UML kullanım durumu ve nesne diyagramları içeren veya içermeyen, benzer büyüklük ve karmaşıklıkta dokümanlar kullanılmıştır. UML diyagramların analist ve kullanıcılar arasındaki iletişim problemlerini azaltmasına benzer şekilde [5], analist ve gözden geçirenler arasındaki iletişim problemlerini de azaltacağını düşünüyoruz.

Çalışmada, gereksinim dokümanına eklenen UML diyagramlarının gereksinim gözden geçirme sürecine okuyucu olarak katılan bireylerin etkenlik ve etkinliklerine olan etkisini inceleyen kontrollü deneyler yapılmıştır. Replikasyonlar bireysel deneylerin bulgularının genelleştirilmesinde kullanılan yararlı çalışmalardır [13, 14, 19]. Yazılım mühendisliği alanının olgunlaşması için daha fazla yakın replikasyon çalışmasının yapılmasını gerekir [17, 18]. Çalışma özgün çalışmaya benzer yakınlıktadır. Özgün çalışmada bildirilen bazı geçerlilik tehditlerini azaltmayı amaçlamaktadır [14, 21]. Yapısal ve içsel geçerlilik problemlerini azaltmak için karmaşıklık ve büyüklük açısından birbirine benzer, özgün çalışmada kullanılandan farklı iki yeni doküman hazırlanmıştır [15]. Gözden geçirme sürecinde okuyucu rolüne sahip kişilerin bireysel performanslarını belirlemede kullanılan dil seviyeleri de önemlidir [23]. Bu nedenle, replikasyon çalışmasında yeni dokümanlar katılımcıların ana dili olan Türkçe'de hazırlanmıştır.

Çalışma sunumunda deneysel replikasyon rehberi izlenmiştir [20]. 2. bölümde geçmişte konu ile ilgili yapılan benzer çalışmalar özetlenmiş, 3. bölümde replikasyon tasarımının ayrıntıları sunulmuştur. Veri analizi bölümünün ardından, 5. bölüm geçerlilik teditlerini, ve 6. bölüm de sonuçlar ve gelecek çalışmalarını anlatmaktadır.

2 İlgili Çalışmalar

UML diyagramlarının gözden geçirme sürecine etkilerini araştıran çalışma sayısı yeterli değildir. Güncel bir literatür taramasında bile benzer çalışma yoktur [24].

Bireysel gözden geçirme sürecinin kalitesini artırmak için araştırmacılar çoklukla okuma teknikleri üstünde odaklanmışlardır [2, 8, 10, 16, 18]. Hangi okuma tekniğinin daha iyi olduğu ile ilgili kesin bir ortak sonuç elde edilmemiş olmasına karşın, kullanım durumları içeren dokümanların etkinliğini artırmak amacıyla yeni bir okuma tekniği olarak Kullanım Tabanlı Okuma (Usage Based Reading-UBR) geliştirilmiştir [8]. UBR önceliklendirilmiş kullanım durumlarını esas alır. Bu çalışmada kullanım durumları diyagramları açıklama içermediğinden, UBR izlenmemiştir. Yazılım gözden geçirme ilgili yapılan çalışmaların önemli bir kısmı gözden geçirme süreci ile ilgilidir. UML diyagramlarının gözden geçirme ile ilgili bir değişken olarak incelenmesi konusu literatürde ele alınmamıştır [2, 3, 10, 21, 23, 33]. Bununla birlikte, yazılım bakım aşaması ile ilgili yapılan çalışmalar deneysel olarak grafik elemanların faydalarını araştırmışlardır [28, 29]. Çalışmaların sonuçlarına göre grafik elemanların varlığı, mimarinin daha iyi anlaşılmasını ve bakım çalışmalarını olumlu etkiler [28]. UML diyagramlarının yararları ile ilgili bilgilere ulaşılabilir [26-30, 36].

UML diyagramlarının gereksinim dokümanlarında yeralmasının raporlanan hata sayısını anlamlı etkilediği bildirilmiştir [21]. Katılımcılar UML diyagramlarının varlığında, daha fazla sayıda hata raporlamışlardır. UML diyagramlarının doğru olarak belirlenen hata sayısına anlamlı bir etkisi bulunmamıştır [21].

Özgün Deney

Özgün çalışmada akademik ortamda, 35 dördüncü sınıf lisans öğrencisinin katıldığı deney yapılmıştır [21]. Her öğrenci iki gereksinim dokümanını incelemiş ve buldukları hataları raporlamışdır. Çalışmada, "bireysel gereksinim gözden geçirme sürecinde UML diyagramlarının gereksinim dokümanında yer alması": H1:"raporlanan hata sayısını", H2:"bulunan hata sayısını", H3:"yanlış pozitif hata sayısını" ve H4:"hata bulmak için gereken zamanı" etkiler hipotezleri incelenmiştir.

Bağımsız değişken: Gereksinim dokümanının UML diyagram içerip içermemesi, bağımlı değişkenler: raporlanan hata sayısı, ve doğru olarak belirlenen hata sayısıdır.

3 Replikasyon Tasarımı

Araştırma Soruları

S1) Bireysel hata raporlama oranı, gereksinim dokümanında UML diyagram kullanılmasından etkilenir mi?

S2) Bireysel hata belirleme oranı, gereksinim dokümanında UML diyagram kullanılmasından etkilenir mi?

S3) Bireysel olarak doğru şekilde belirlenen önem seviyesi yüksek hata sayısı, gereksinim dokümanında UML diyagram kullanılmasından ile etkilenir mi?

S4) Bireysel olarak kaydedilen yanlış pozitif sayısı, gereksinim dokümanında UML diyagram kullanılmasından etkilenir mi?

S5) Gereksinim hatasının bulunma zamanı, gereksinim dokümanında UML diyagram kullanılmasından etkilenir mi?

Değişkenler

Bağımsız değişken, HasUML, gereksinim dokümanının UML diyagram içerdiği durumda 1, içermediği durumda 0 değerini alır. Bağımlı değişken tanımları:

Raporlanan hata sayısı (RD): Her bir katılımcı tarafından raporlanan toplam hata sayısı

Doğru olarak belirlenen hata sayısı (CDD): Bireysel gözden geçiren tarafından doğru olarak bulunan toplam hata sayısı, etkenlik ölçüsü

Önemli hata sayısı (HSD): Bireysel gözden geçiren tarafından doğru olarak bulunan diğer hata türü olmayan, toplam hata sayısı, etkenlik ölçüsü

Yanlış pozitif sayısı (FP): Bireysel gözden geçiren tarafından kaydedilen hatalı yanıtlar. Bu değişkenin yüksek değer alması istenmez.

Hata belirleme zamanı (AT): Doküman başına gözden geçirme için harcanan zaman. Bu çalışmada zaman değişkeni göz önüne alınmıştır [27].

Hipotezler

Yerden kazanmak için sıfır hipotezleri aşağıdaki hipotezler listesine eklenmedi:

H1: Gereksinim dokümanında bulunan UML diyagramlarının varlığı raporlanan hata sayısını (RD) etkiler.

H2: Gereksinim dokümanında bulunan UML diyagramlarının varlığı doğru bulunan hata sayısını (CDD) etkiler.

H3: Gereksinim dokümanında bulunan UML diyagramlarının varlığı bulunan önemli hata sayısını (HSD) etkiler.

H4: Gereksinim dokümanında bulunan UML diyagramların varlığı yanlış pozitif sayısını (FP) etkiler.

H5: Gereksinim dokümanında bulunan UML diyagramların varlığı doküman gözden geçirme süresini (AT) etkiler.

Katılımcılar

Katılımcılar özel bir üniversitenin lisans programında zorunlu bir ders olan Uygulamalı Yazılım Mühendisliği dersine kayıtlı 66 adet dördüncü sınıf öğrencisidir. Tümü gönüllü olarak katılmıştır. Katılımcıların iş deneyimi ortalama 4 aydır. Gözden geçirme verilen eğitimle sağlanmıştır. Öğrenciler UML diyagramı çizme ve yazılım gereksinim dokümanı hazırlama konusunda başlangıç seviyesinde bilgi sahibidir.

Deney Hazırlığı ve Uygulaması

Deney öncesinde 75 öğrenciye gereksinim dokümanı gözden geçirme süreci ile ilgili olarak 50 dakika süren bir eğitim verilmiştir. Eğitimde okuma teknikleri hakkında bilgi verilmiş ve deneyle ilgili açıklamalar yapılmıştır. Deneye katılacak grupların

oluşturulmasında katılımcıların UML ile ilgili ön testte ve akademik performans değerleri ve deneyimleri gözönüne alınmıştır. Deneye 66 öğrenci katılmıştır.

Deney iki aşamada gerçekleştirilmiştir. Her bir aşama 30 dakika ile sınırlandırılmıştır. Çalışmanın toplam süresi, eğitimi de içerecek şekilde, 110 dakika olmuştur. Deneyin tasarımı Tablo 1'de gösterilmektedir. Her aşama sonrasında materyaller toplanmış yenileri dağıtılmıştır.

Tablo 1. Deney Aşamaları

Eğitim ve Ön test: 50 dakika			
Aşama I: 30 dakika		***Diyagram Tipi***	
		UML	No-UML
Sistem	A: PARSİM	Grup 1	Grup 2
	B: QFD	Grup 4	Grup 3
Aşama II: 30 dakika		***Diyagram Tipi***	
		UML	No-UML
Sistem	A: PARSİM	Grup 3	Grup 4
	B: QFD	Grup 2	Grup 1

Kullanılan Materyal

Çalışmanın ön test kısmında JORGIS ve eğitim kısmında (Cause Effect Graphing Tool) CEGT isimli farklı gereksinim dokümanları kullanılmıştır. Deney için güncellenmiş B: Quality Function Deployment (QFD) ve A: Particle Simulator Software (ParSIM) dokümanları kullanılmıştır. Katılımcılar dokümanların ilgili olduğu konulara tanıdık değillerdi. Her bir sistem için UML diyagramları içeren ve içermeyen sürümleri olmak üzere ikişer gereksinim dokümanı oluşturulmuş, UML diyagramlarının hiçbiri için ayrıntılı açıklama verilmemiş, yalnızca diyagramlar eklenmiştir. Kullanılan sistemler karmaşıklık ve boyut açısından benzerdir ve ilgili standarda [35] uyumludur.

Deneyde kullanılan materyaller, sistem yazılım gereksinim dokümanı, dokümanda bulunan hataların belirlenmesinde kullanılacak kontrol listesi, ve hataların kaydedilmesinde kullanılacak hata kayıt formundan oluşmaktadır. Katılımcıların deney sonrasındaki gözlem ve yorumlarını almak için deney sonrasında bir anket yapılmıştır. Dokümanlara serpiştirilen hata türleri: eksiklik, belirsizlik, tutarsız bilgi, hatalı bilgi, gereksiz bilgi ve diğer olmak üzere altı sınıftan oluşmaktadır [10].

Tasarım

Deney gerçekleştirilmeden önce katılımcılara UML bilgilerini ölçen bir ön-test uygulanmış ve gözden geçirme süreci ile ilgili olarak 50 dakika eğitim verilmiştir. Her bir katılımcının iki ayrı sistem gereksinim dokümanını gözden geçirmesini mümkün kılmak amacıyla 4 ayrı grup oluşturulmuştur. Sistemlerin UML içeren ve içermeyen versiyonları ile toplam 4 adet gereksinim dokümanı Tablo 1'deki gibi atanmıştır. Böylece her bir katılımcı hem UML içeren ve içermeyen iki ayrı dokümanı gözden geçirebilmiştir. Tasarım oluşabilecek öğrenme etkisini de azaltmıştır.

4 Veri Analizi ve Sonuçlar

Deney gruplarında yeralan öğrencilerin dağılımı not ortalamaları ve ön testten aldıkları sonuçlara göre başlangıçta homojen olarak dağıtılmışlardı. Not ortalamalarını (CGPA) genel akademik başarısı ve ön test sonuçlarını UML bilgilerini ölçme amacıyla kullandık. Bu değerleri kullanarak 4 grup oluşturuldu. Ancak, eğitime katılan dokuz öğrencinin deneye katılmaması nedeniyle, gruplar arasında planlanan dengeden sapma oluştu. Tablo 2 RD değişkeni için betimsel istatistik değerlerini göstermektedir. Deneye katılan 66 katılımcı ANOVA varsayımlarını sağladı [19, 24, 32].

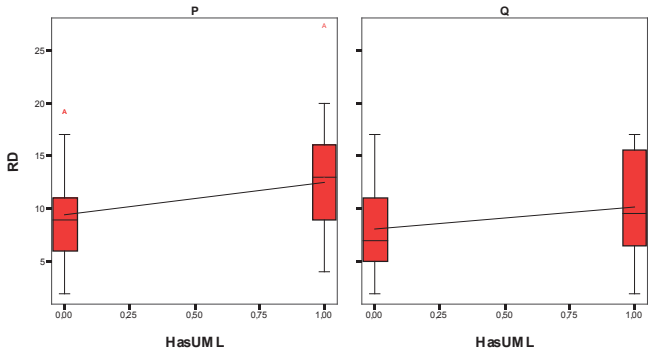
Tablo 2. RD Örneklem Tanımı, N=131

	Min	Max	Averaj	Sapma		Min	Max	Averaj	Sapma
RD	2,00	27	10,03	4,894	**HSD-**	0	10	2,69	2,281
CDD	0	17	6,74	3,778	**FP-**	0	20	3,29	3,529

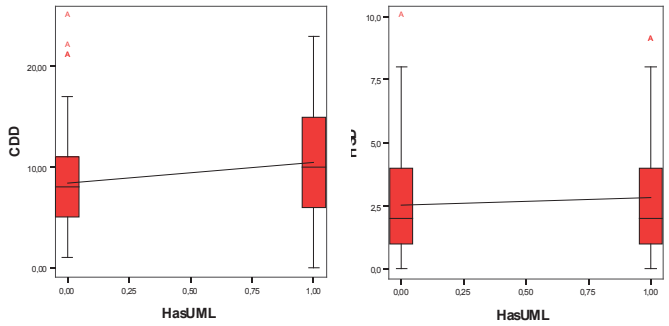
Kullanılan doküman UML diyagram içerdiğinde hacmi artmasına karşın, katılımcılar dokümanın UML içermediği duruma oranla daha fazla hata raporlamışlardır (Şekil 1). Sadece RD değil, aynı zamanda CDD değişkeni için de UML diyagramı içeren dokümanlar incelendiğinde daha fazla sayıda doğru hata tespit edilmiştir (Şekil 2). Tespit edilen tüm hataların önem derecesi aynı değildir [25]. Çalışmada "diğer" hata tipi altında, genellikle yazım ile ilgili hataların, öteki hata tiplerine nazaran "daha az önemli" oldukları varsayıldı. Bu nedenle HSD değeri hesaplanırken, "diğer" türüne giren hatalar kapsam dışında tutulmuştur.

Not ortalamaları bazlı oluşturulan her bir grup, içinde UML diyagram bulunan dokümanda daha fazla hata olduğunu raporlamıştır (Şekil 3). UML diyagramlarının raporlanan hata sayısına olan katkısı yüksek akademik ortalama değerine sahip olan öğrenciler arasında en azdır. İkinci gözden geçirmede daha az sayıda hata raporlanmıştır (Şekil 4).

Çalışmada yeralan tüm istatistik testlerde alfa düzeyi 0.05 kullanıldı. Katılımcılar arasındaki testler dokümanların incelenme sırası ve akademik ortalama grubu değişkenlerinin RD değişkeni üzerinde anlamlı etkisi olduğunu gösterdi (Tablo 3 ve Tablo 4). ANOVA test sonuçlarını anlatan tablolarda şu kısaltmalar kullanılmıştır: KT: Kareler Toplamı, m: Ortalama, p: anlamlılık derecesi, GA: Gruplar Arası, Gİ: Grup İçi, T: Toplam



Şekil 1.HasUML ve İncelenen Doküman Bazında RD



Şekil 2. HasUML Bazında CDD ve HSD

ANOVA test sonuçları doküman inceleme sırası ((F129,1=31,233, p=0,000) ve akademik not ortalaması grubu (F118,3=2,776, p=0,044) değişkenlerinin RD ile anlamlı ilişkisi olduğunu göstermiştir. Doküman inceleme sırası ve akademik performans değişkenleri ilişkili değildir.

RD(Raporlanan hata sayısı): ANOVA testinin sonuçları HasUML değişkeninin raporlanan hata sayısı değişkeni üstünde anlamlı bir etkisi olduğunu göstermiştir (F129,1=10,347, p=0,002). Katılımcılar gözden geçirilen doküman UML diyagramlar içerdiğinde daha fazla sayıda hata raporlamışlardır (Tablo 5).

CDD(Doğru belirlenen hata sayısı): ANOVA testinin sonuçlar HasUML değişkeni ile CDD değişkeni arasında anlamlı bir ilişki olduğunu bildirmektedir. Katılımcılar incelenen dokümanların UML diyagram içermesi durumunda daha fazla sayıda doğru hatayı belirlemişlerdir (F129,1=7,253, p=0,008) (Tablo 7).

HSD(Bulunan önemli hata sayısı): ANOVA testinin sonuçları HasUML değişkeninin HSD değişkeni üstünde anlamlı bir ilişkisi olmadığını göstermektedir (Tablo 8). Anlamlı olmayan bu ilişkinin nedeni belirlenen az önem derecesine sahip hataların kaydedilmesi nedeniyle diğer hataların tespit edilmesi için kalan zamanın sınırlı olması olabilir. Katılımcılar UML diyagram içeren dokümanlar incelediklerinde UML içermeyen dokümana nazaran daha fazla sayıda önemli hata belirlemişlerdir (Şekil 5).

FP(Yanlış Pozitif): ANOVA test sonuçları HasUML değişkeninin FP üzerinde anlamlı bir etkisi olmadığını önermiştir FP (F129,1=2,250, p=0,136). (Tablo 9). Katılımcılar UML diyagramlar dahil edildiğinde daha fazla sayıda FP raporlamışlardır (Şekil 6).

AT(Harcanan Zaman): Gözden geçirme sürecinde harcanması için belirlenen zamanın üst limiti olmasına karşın, ANOVA test sonuçları UML diyagramların dahil edilmesinin gözden geçirme sürecinde harcanan zaman üstünde pozitif ve anlamlı bir etkisi olduğunu önermiştir (F126,1=25,564, 0,000) (Tablo 10). (Şekil 7).

Tablo 3. RD ve CGPA-Group ANOVA

	KT	*df*	*m*	*F*	*p*
GA	190,125	3	63,375	2,776	,044
Gİ	2625,152	115	22,827		
T	2815,277	118			

Tablo 4. RD ve CEGT-Total ANOVA

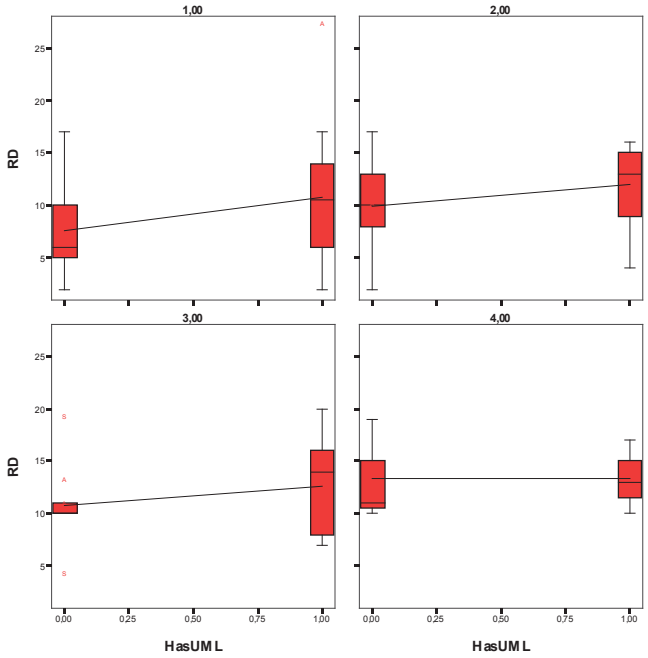
	KT	*df*	*m*	*F*	*p*
GA	95,696	5	19,139	,793	,557
Gİ	3018,182	125	24,145		
T	3113,878	130			

Tablo 5. RD ve HasUML ANOVA

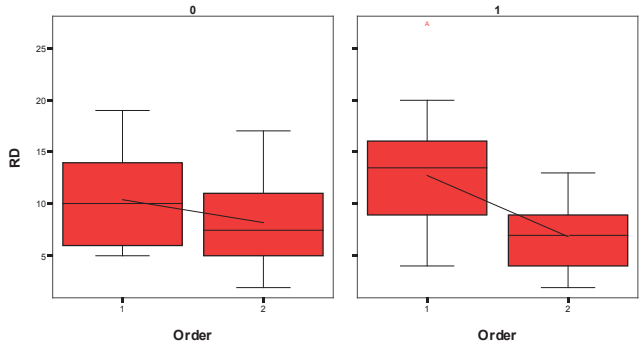
	KT	*df*	*m*	*F*	*p*
GA	231,209	1	231,209	10,347	,002
Gİ	2882,669	129	22,346		
T	3113,878	130			

Tablo 6. RD ve Sıra ANOVA

	KT	*df*	*m*	*F*	*p*
GA	608,957	1	806,957	31,233	,000
Gİ	2506,921	129	19,433		
T	3113,878	130			



Şekil 3. Akademik Performans Grupları ve HasUML Değişkenleri Bazında RD



Şekil 4. Sıra(Order) ve HasUML Bazında RD

Deney sonrası yapılan ankete göre kullanılan materyal ne zor ne de kolaydır, süre uygundur. Katılımcılar UML diyagramların gözden geçirme sürecinde faydalı olduklarını belirtmelerine karşın, diyagramları ayrıntılı olarak incelememişlerdir.

Tablo 7. CDD ANOVA

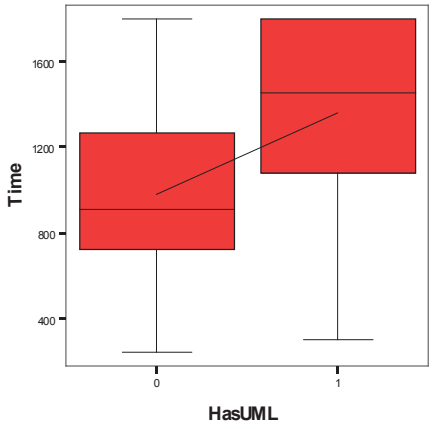
	KT	*df*	*m*	*F*	*p*
GA	98,761	1	98,761	7,253	,008
Gİ	1756,415	129	13,616		
T	1855,176	130			

Tablo 8. HSD ANOVA

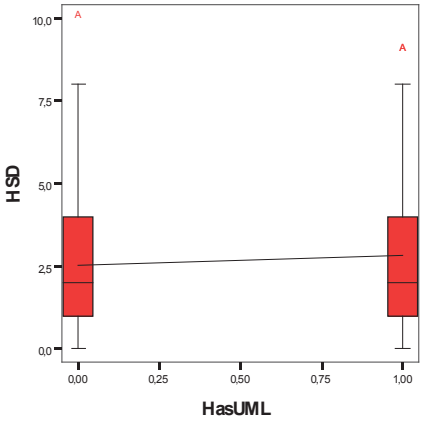
	KT	*df*	*m*	*F*	*p*
GA	3,267	1	3,267	,626	,430
Gİ	672,901	129	5,216		
T	676,168	130			

Tablo 9. FP ANOVA

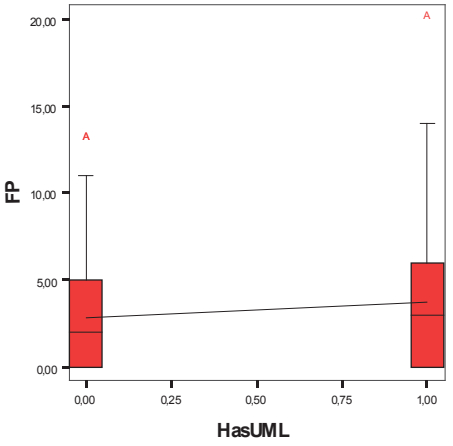
	KT	*df*	*m*	*F*	*p*
GA	27,749	1	27,749	2,250	,136
Gİ	1591,228	129	12,335		
T	1618,977	130			



Şekil 7. HasUML Bazında AT



Şekil 5. HasUML Bazında HSD



Şekil 6. HasUML Bazında FP

TABLO 10. AT ANOVA

	KT	*df*	*m*	*F*	*p*
GA	4573363	1	4573363,200	25,564	,000
Gİ	22540886	126	178895,924		
T	27114250	127			

Çalışma Sonuçlarının Karşılaştırılması

Özgün çalışmada olduğu gibi UML diyagramlarının gereksinim dokümanına eklenmesinin raporlanan hata sayısı üstünde anlamlı ve pozitif etkisi gözlemlenmiştir.

Özgün çalışmada HasUML'in CDD değişkeni üstünde anlamlı bir etkisi bulunmamıştır, replikasyonda ise HasUML ile CDD arasında anlamlı ilişki gözlemlenmiştir.

Replikasyon çalışmasında tanımlanan HSD ve FP değişkenleri üstünde HasUML değişkeninin anlamlı etkisi gözlemlenmemiştir. Öte yandan HasUML değişkeninin AT değişkeni üstünde pozitif ve anlamlı etkisi vardır.

Akademik not ortalaması bazında tüm gruplar UML diyagramı içeren dokümanları kullandıklarında daha fazla sayıda hata raporlamışlardır. Raporlanan hata sayıları arasındaki fark akademik performansı en düşük olan grupta en yüksektir. Akademik performansı en yüksek olan grupta ise bu fark minimumdur.

5 Geçerlilik Tehditleri

İç geçerlilik bir çalışmanın sonuçlarının tasarımındaki hatalara değil de manipüle edilen bağımsız değişkenlere bağlanabilme derecesi, dış geçerlilik genelleştirme ile ilgilidir [34]. Yazılım mühendisliği ile ilgili çalışmalarda öğrencilerin katılımcı olmaları yaygındır [31]. Öğrencilerin katılmasının ciddi bir tehdit olmadığını düşünüyoruz. Katılımcılar alan bilgisi yerine UML modelleme bilgisine tanıdık bireylerdir. Özgün deneyle replikasyon çalışması arasındaki en önemli farkın (iyileştirmenin) kullanılan materyaller (büyüklük ve karmaşıklık), katılımcıların alan bilgisine olan yakınlıkları, ve deney öncesinde verilen eğitim, ile deney sonrası ve öncesi yapılan testler olduğunu bildirmeliyiz. Replikasyon çalışmasındaki materyaller özgün çalışmadakinden farklıdır. Farklı materyallerin kullanılması dış geçerliliğinin artırılmasında yardımcıdır.

Okuma tekniği gözden geçirme sürecinin etkinliğinde önemli bir değişkendir [2, 3, 8, 10]. Farklı bir okuma tekniğinin kullanılması geçerlilik tehditlerini azaltacaktır.

Çalışma uygulamasında deneyi tasarlayan ve sonuçlarını değerlendiren yazar dışında, çalışmayla doğrudan ilgisi olmayan gönüllü iki farklı öğretim elemanının görev alması deneyi yapan kişi önyargısını azaltmıştır. Öğretim elemanları ve katılımcılar öğrenciler çalışmanın hipotezleri ve araştırma soruları hakkında bilgi sahibi değildirler. Yapısal geçerlik ile ilgili olarak ikinci aşamada öğrencilerin bir kısmının deneyi erken tamamlama eğilimde olması gözlemlenmiştir.

6 Sonuç ve Gelecek Çalışmalar

Çalışma sonuçları özgün çalışmada raporlanan hata sayısı değişkeni ile ilgili bulguları desteklerken, doğru belirlenen hata sayısı değişkeni ile ilgili bulgularla çelişmektedir. Doküman hacminde diyagramların dahil edilmesiyle artışa karşın, aynı zamanda bulunan doğru hata sayısı diyagramların dahil edildiği durumlarda daha fazladır. Replikasyonda yeni değişkenler, hata tipleri önem dereceleri, yanlış hata sayısı ve kullanılan zaman da incelenmiştir. Çalışmada gözden geçirenlerin etkinlikleri incelenen doküman UML içerdiğinde anlamlı olarak artarken, verimlilikleri anlamlı olarak azalmıştır. Katılımcılar UML içeren doküman incelemelerinde daha fazla zaman harcamış ve daha fazla sayıda hata raporlamışlardır. Etkinlik artımı UML'lerin gereklerin anlaşılmalarına olan katkıları ile ilişkilendirilebilirken, verimlilikteki azalma da di-

yagramların okunması için harcanan ek zaman kullanımı ile açıklanabilir. Kesin ve genelleştirebilir sonuçlar elde edilmesi için daha fazla sayıda ve farklı ortamlarda replikasyon çalışmalarının yapılması gerekmektedir. Çalışmada kullanılan materyallerin Türkçe ve İngilizce versiyonları için yazar ile bağlantıya geçilmesini öneriyoruz.

7 Teşekkür

Marcela Genero Bocco, Erhan Yüceer ve Tuna Kılıç'a yorumları, Duygu Albayrak, Syed A. Ali ve katılımcılara çalışmanın gerçekleşmesindeki katkıları için teşekkürler.

8 Kaynaklar

[1] S. Engineering, S. Committee, and I. Computer, IEEE Std 1028TM-2008 (Revision of IEEE Std 1028-1997), IEEE Standard for Software Reviews and Audits, vol. 2008, no. August. 2008.

[2] A. Aurum, H. Petersson, and C. Wohlin, 2002. "Stateof- the-art: Software Inspections After 25 Years," Software Testing, Verification and Reliability.vol. 12, 2002, pp.133-154.

[3] A. Porter, L. Votta, V. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment," IEEE Trans. on Software Engineering, vol. 21, 1995, pp.563-575.

[4] Booch, G., J. Rumbaugh, and I;. Jacobson, The Unified Modeling User Guide. Addison Wesley, 1999.

[5] A. Endres and D. Rombach, A Handbook of Software and System Engineering: Empirical Observations, Laws and Theories, IESE, 2003.

[6] S. N. Bhatti, "Why Quality? ISO 9126 Software Quality Metrics (Functionality) Support by UML Suite," *Software Engineering Notes*, vol. 30, no. 2, pp. 1-5, 2005.

[7] G. Kösters, Hans-Werner S., and M. Winter, "Coupling Use Cases and Class Models as a Means for Validation and Verification of Requirements Specifications," *Requirements Engineering*, vol. 6, no. 1, pp. 3-17, Feb. 2001.

[8] Thelin, T., Runeson, P. Wohlin, C., "Prioritized Use Cases as a Vehicle for Software Inspections", IEEE Software, vol 20, no.4, 30-33, 2003.

[9] F. Salger, G. Engels, and A. Hofmann, "Inspection effectiveness for different quality attributes of software requirement specifications: An industrial case study," *2009 ICSE Workshop on Software Quality*, pp. 15-21, May. 2009.

[10] J. C. Carver, N. Nagappan, and A. Page, "The Impact of Educational Background on the Effectiveness of Requirements Inspections: An Empirical Study," vol. 34, no. 6, pp. 800-812, 2008.

[11] R. Conradi, P. Mohagheghi, T. Arif, L. C. Hegde, G. A. Bunde, and A. Pedersen, "Object-Oriented Reading Techniques for Inspection of UML Models – An Industrial Experiment," ECCOP pp. 483-500, 2003.

[12] K. Cox and` R. Jeffery, "An Experiment in Inspecting the Quality of Use Case Descriptions," *Practice*, vol. 36, no. 4, pp. 211-229, 2004.

[13] Basili V., Shull F. and Lanubile F. (1999) Building knowledge through families of experiments, IEEE Transactions on Software Engineering, 25(4), 435-437, 1999.

[14] F. Shull, J. Carver, S. Vegas, N. Juristo. The Role of Replications in Empirical Software Engineering. *Empirical Software Engineering Journal*. 13:211–218. 2008.

[15] Juristo, N. and Vegas, S. Using differences among replications of software engineering experiments to gain knowledge. In Anonymous *ESEM '09: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement.* (). IEEE Computer Society, Washington, DC, USA, 356-366, 2009.

[16] B. Kitchenham. The role of replications in empirical software engineering—a word of warning. *Empirical Software Engineering Journal*. 13:219–221. 2008.

[17] Vegas, S., Juristo, N., Moreno, A.M., Solari, M., and Letelier, P. 2006. Analysis of the Influence of Communication between Researchers on Experiment Replication. ISESE 2006. 28-37.

[18] F. Shull, J. Carver,G.H. Travassos, J.C. Maldonado, R. Conradi and V.R. Basili, Replication Studies: Building a Body of Knowledge About Software Reading Techniques, in Lecture Notes on Empirical Software Engineering eds. N. Juristo and A. M. Moreno, 39-84, 2003.

[19] Cruz-Lemus, J. a, Genero, M., Manso, M. E., Morasca, S., & Piattini, M. (2009). "Assessing the understandability of UML statechart diagrams with composite states—A family of empirical studies". *Empirical Software Engineering*, *14*(6), 685-719. doi:10.1007/s10664-009-9106-z

[20] Carver, J. C. "Towards reporting guidelines for experimental replications: A proposal." In Proceedings of the 1st International Workshop on Replications in Empirical Software Engineering (Held during ICSE). May 4, 2010.

http://cs.ua.edu/~carver/Papers/Conference/2010/2010_RESER.pdf

[21] Ö. Albayrak, "An Experiment to Observe the Impact of UML Diagrams on the Effectiveness of Software Requirements Inspections," ESEM 2009, pp. 506-510.

[22] M. Genero, and M. Piattini, "Empirical validation of measures for class diagram structural complexity through controlled experiments", 5th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2001.

[23] Carver, J. C. (2003) PhD. Thesis. The Impact of Background and Experience on Software Inspections. University of Maryland, 2003.

[24] Mohagheghi, P., Dehlen, V., & Neple, T. (2009). Definitions and approaches to model quality in model-based software development – A review of literature. *Information and Software Technology*, *51*(12), 1646-1669. Elsevier B.V. doi:10.1016/j.infsof.2009.04.004

[25] Laitenberger, O., Atkison, C., El-Emam, K. "Using Inspection Technology in Object-oriented Development Projects," NRC/ERB-1077, no. June, 2000.

[26] Bratthall, L., & Wohlin, C. (2002). Is it possible to decorate graphical software design and architecture models with qualitative Information?-An experiment. IEEE Transactions on Software Engineering, 28(12), 1181-1193. doi:10.1109/TSE.2002.1158290

[27] Ricca, F., Penta, M. D., Torchiano, M., Society, I. C., Tonella, P., & Ceccato, M. (2010). How Developers' Experience and Ability Influence Web Application Comprehension Tasks Supported by UML Stereotypes: A Series of Four Experiments. Computer, 36(1), 96-118.

[28] M. Staron, L. Kuzniarz, and C. Thurn. An Empirical Assessment of Using Stereotypes to Improve Reading Techniques in Software Inspections. in 3-WoSQ: Third Workshop on Software Quality.2005. St. Louis, USA: ACM.

[29] Arisholm, E., Briand, L. C., Member, S., Hove, S. E., & Labiche, Y. (2006). The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation, *IEEE Trans. On Software Engineering* 32(6), 365-381.

[30] Dzidek, W. J., Arisholm, E., & Briand, L. C. (2008). A realistic empirical evaluation of the costs and benefits of UML in software maintenance. *IEEE Transactions on Software Engineering*, *34*(3), 407-432.

[31] Carver, J., Jaccheri, L., Morasca, S., Shull, F., Software, E., Group, E., & Science, I. (n.d.). Using Empirical Studies during Software Courses, 81-103. 2004 ESERNET 2001-2003LNCS 2765

[32] F.J. Gravetter and L.B. Wallnau, Statistics for the Behavioural Sciences, 7th edition, 2007.

[33] D.E. Harter, C.F. Kemerer, and S.A. Slaughter, "Does Software Process Improvement Reduce The Severity of Defects? A Longitudınal Field Study", IEEE Trans. On Software Engineering, accepted paper.

[34] Wohlin, C., P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers, 1999.

[35] IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, 25 August 1998.

[36] Kılıç, Ö., Say, B., Demirörs, O. Cognitive Aspects of Error Finding on a Simulation Conceptual Modeling Notation, Computer and Information Sciences, ISCIS 2008, pp.1-6.