BŞEÜ Fen Bilimleri Dergisi
7(2), 825-836, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.699266

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Araştırma Makalesi - Research Article

WEKA Veri Madenciliği Yazılımının Sürümleri Arasındaki Kalite Değişimlerinin QMOOD ile İncelenmesi

Hakan Gündüz[1*]

Geliş / Received: 19/03/2020 ***Revize / Revised: 01/07/2020*** ***Kabul / Accepted: 05/07/2020***

ÖZ

QMOOD (Quality Model for Object Oriented Design), dört katmandan oluşan ve bu katmanlar arasındaki ilişkileri değerlendiren hiyerarşik yapılı bir tasarım kalite modelidir. Bu model nesneye dayalı yazılım metriklerini kullanarak yazılım kalite niteliklerinin değerlerini hesaplar. Bu çalışmada, QMOOD kullanılarak, açık kaynak kodlu WEKA veri madenciliği yazılımı sürümlerinin kalite değişimleri gözlenmiştir. Yazılıma yeni sürümlerde farklı özelliklerin eklenmesi ve yazılım tasarım yapısının değişmesi QMOOD'un işlevsellik, esneklik ve yeniden kullanılabilirlik gibi niteliklerini doğrudan etkilerken, sürümlerin kalıtım hiyerarşisi değişikliği ise genişletilebilirlik ve etkinlik niteliklerinin puanlarında oynaklığa sebep olmuştur. Anlaşılırlık niteliğinin değerini ise yeni sürümlerde artan metot ve sınıf sayısı olumsuz yönde etkilemiştir. Çalışmanın sonucunda QMOOD ile elde edilen kalite puanlarıyla WEKA sürümlerindeki yapısal değişimlerin paralel olduğu gözlenmiştir.

Anahtar Kelimeler- Yazılım Tasarım Kalitesi, QMOOD, WEKA

[1]*Sorumlu yazar iletişim: h.gunduz@bandirma.edu.tr (https://orcid.org/0000-0003-2152-5490)
Yazılım Mühendisliği Bölümü, Bandırma Onyedi Eylül Üniversitesi, Bandırma, Balıkesir, Türkiye

BŞEÜ Fen Bilimleri Dergisi
7(2), 825-836, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.699266

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

Investigation of Quality Changes between Versions of WEKA Data Mining Software Using QMOOD

ABSTRACT

QMOOD (Quality Model for Object Oriented Design) is a hierarchical design quality model consisting of four layers and evaluates the relationships between these layers. This model calculates the values of software quality attributes using object-oriented software metrics. In this study, quality changes of open source WEKA data mining software versions were observed using QMOOD. While adding new features to the software and changing the software design structure directly affected the attributes of QMOOD, such as functionality, flexibility, and reusability, the hierarchy change of the versions caused volatility in the scores of extensibility and effectiveness. On the other hand, the increasing number of methods and classes in new versions negatively affected the value of understandability. As a result of the study, it was observed that the structural changes in the WEKA versions were parallel with the quality scores obtained with QMOOD.

Keywords- Software Design Quality, QMOOD, WEKA

BŞEÜ Fen Bilimleri Dergisi
7(2), 825-836, 2020

BSEU Journal of Science
DOI: 10.35193/bseufbd.699266

e-ISSN: 2458-7575 (http://dergipark.gov.tr/bseufbd)

I. GİRİŞ

Yazılım tasarımı, yazılım geliştirme sürecinin yaratıcı ve en anlamlı safhasıdır ve yazılım kalitesinin geliştirilmesi ve kontrolünde önemli rol oynar. Yazılım tasarımının kalitesi doğrudan yazılım ürününü kalitesini etkilemektedir. Yazılım kalitesi, yazılımla ilgili paydaşların gereksinimlerinin ne kadar karşılandığının ölçüsüdür. Yazılım kalitesinin ölçülüp değerlendirilmesinden önce kalite modelinin oluşturulması gerekir. Yazılım dünyasında kullanılan kalite modelleri, yazılım kalitesini etkileyen kalite bileşenlerini (niteliklerini) ve aralarındaki ilişkileri tanımlamaktadır. Kalite modelinde yazılımın kalitesi karakteristikler ve varsa alt karakteristikler ile belirlenmektedir. Yazılım kalitesinin ölçülmesinde yazılımın nicelik ile ifade edilebilen basit özellikleri kullanılmaktadır. Kalite modelleri aynı zamanda sektördeki yazılımcılar arasında ortak bir dil oluşturmaktadır.

Bu çalışmada Weka veri madenciliği yazılımının sürümleri arasındaki kalite değişimleri incelenmiştir. Weka, Java programlama dilinde yazılmış olup nesneye dayalı tasarıma sahiptir [1]. Weka yazılımının sürümlerinin kalitesinin belirlemede QMOOD modeline benzer şekilde katmanlı bir yapı oluşturulmuştur [2]. Oluşturulan modelde QMOOD modelinde yer alan tasarım kalitesi nitelikleri seçilmiştir. Model de kullanılan nesneye dayalı metrikler ise iPlasma aracıyla çıkarılmış ve nesneye dayalı tasarım özelliklerinin değerlendirilmesinde kullanılmıştır [3]. Nesneye dayalı tasarım özellikleri ile tasarım kalitesi arasındaki ilişkilere bakılarak her bir sürümün kalitesi belirlenmiş ve sürümler arası kalite değişimleri bulunmuştur.

II. GEÇMİŞ ÇALIŞMALAR

Yazılım kalitesini ölçmek ve değerlendirmek için ilk olarak bir kalite modeli oluşturularak ölçülmek istenenler ve kaliteyi oluşturan bileşenler net bir şekilde tanımlanır. İkinci adımda ölçme yöntemine karar verilir buna uygun olarak veri toplanır. Son adım da ise toplanan veriler (metrikler) ile kalite arasındaki ilişkiler tanımlanır ve yapılan ölçme değerlendirilir.

Literatürde yazılım kalitesini değerlendirmek için oluşturulan modellerin [4-7] yanı sıra nesne yönelimli programlamanın önem kazanmasıyla birlikte nesne yönelimli tasarım kalitesini değerlendirmek amacıyla hiyerarşik modeller de tanımlanmıştır [2]. 1977 yılında McCall tarafından oluşturulan yazılım modeli pek çok yazılımın temeli olarak kabul edilir [8]. Oluşturulan model kullanıcılar ile geliştiriciler arasındaki iletişimi sağlamayı amaçlar. 1992 yılında Victor Basili tarafından oluşturulan “Goal/Question/Metric-GQM” yaklaşımı ise (Hedef/Soru/Metrik) ilk olarak projelerde ortaya çıkan hataları değerlendirilmek için oluşturulmuş fakat daha sonra kalite iyileştirme kavramına uygun hale getirilmiştir [9, 10]. 1991 yılında ISO/IEC yazılım kalitesi tanımlarının standartlaştırılması amacıyla 4 bölümlük bir doküman oluşturmuştur [11, 12].

Nesneye dayalı yazılım tasarımı kalitesinin ölçülmesi için birçok nesneye dayalı metrik seti de geliştirilmiştir. Chidamber ve Kemerer, nesne tabanlı tasarımlar için küçük metrik seti geliştirmiştir. Bu metrik seti 6 metrik içermektedir [13]. Diğer geliştirilen metrik setleri Lorenz ve Kidd Metrikleri [14], Li ve Henry Metrikleri [15], Henderson-Sellers [16]’dir.

Birçok çalışmada MOOD ve QMOOD metrikleri kullanılmıştır [17]. MOOD metrikleri ise yapıyla ilgili basit özellikler içermektedir. Bunlar encapsulation (kapsülleme) (Method Hiding Factor (MHF) ve Attribute Hiding Factor (AHF)), inheritance (kalıtım) (Method Inheritance Factor (MIF) ve Attribute Inheritance Factor (AIF)), polymorphism (çokşekillilik) (Polymorphism Factor (PF)), message passing (Coupling Factor (CF))’dir [18].

Bansiya ve Davis, Quality Model for Object Oriented Design (Nesne Tabanlı Tasarım için Kalite Modeli) (QMOOD) metriklerini tanımlamıştır. QMOOD, yeniden kullanılabilirlik (reusability), işlevsellik (functionality), etkinlik (effectiveness), anlaşılırlık (understandability), genişletilebilirlik (extendibility), esneklik (flexibility) gibi tasarım kalite niteliklerini değerlendirebilen güçlü bir modeldir. Nitelikler için kalite değerlendirilmesi temel alınarak yazılım sisteminin toplam kalite indeksi (Total Quality Index (TQI)) hesaplanmaktadır [2]. QMOOD kullanılarak yazılım tasarım kalitesi ölçülürken metriklerin değer aralıkları farklı olduğundan, z-puanı normalizasyonu kullanılmaktadır [19].

Birden fazla metrik setinin kullanıldığı çalışmalarda bulunmaktadır [20]. Chidamber-Kemerer (CK) ve Lorenze-Kidd (LK) metrik kümeleri kullanılarak Java programlarının kalitesi ölçülmüştür. Çalışmada bu iki metrik kümesi ağırlıklandırılarak kullanılmıştır [21]. Jiang ise kod seviyesinde metriklerin tasarım seviyesindeki metriklerinden daha iyi performans ortaya konduğunu göstermiş ve iki seviyedeki metriklerin kullanılmasıyla en iyi sonucun alındığını bulmuştur [22]. Subramanyam ve arkadaşları ise CK metrik setini kullanarak Java ve C++ dilinde yazılmış yazılımların hatalarını bulmuş ve sonuçların programlama dillerine göre değiştiğini göstermiştir [19].

III. WEKA VERİ MADENCİLİĞİ YAZILIMI

Weka, makine öğrenimi amacıyla Waikato Üniversitesinde geliştirilmiş ve "Waikato Environment for Knowledge Analysis" kelimelerinin baş harflerinden oluşmuş yazılımın ismidir. Günümüzde yaygın kullanımı olan çoğu makine öğrenmesi algoritmalarını ve metotlarını içermektedir. Java dilinde geliştirilmiş olması ve kütüphanelerinin jar dosyaları halinde geliyor olması sayesinde, JAVA dilinde yazılan projelere kolayca entegre edilebilmesi kullanımını daha da yaygınlaştırmıştır. Yazılım, GNU Genel Kamu Lisansı ile dağıtılmaktadır [23].

Weka, tamamen modüler bir tasarıma sahip olup, içerdiği özelliklerle veri kümeleri üzerinde görselleştirme, veri analizi, iş zekâsı uygulamaları, veri madenciliği gibi işlemler yapabilmektedir. Weka yazılımı, kendisine özgü olarak bir arff uzantısı desteği ile gelmektedir. Ancak Weka yazılımının içerisinde CSV dosyalarını da ARFF formatına çevirmeye yarayan araçlar mevcuttur. Weka ile sınıflandırma, kümeleme ve ilişkilendirme gibi temel veri madenciliği görevleri gerçekleştirilebilir. Bu görevlere ek olarak, veri kümelerinin önişlenmesi ve görselleştirilmesi gibi işlemler de Weka ile yapılabilir. Çalışmada Weka'nın 3.7 geliştirici sürümü kullanılmıştır. Bu sürüm 12 alt sürüme sahiptir.

IV. HAZIRLANAN MODEL VE KULLANILAN METRİKLER

Weka 3.7'nin sürümler arası kalite değişimini değerlendirebilmek için QMOOD' da olduğu gibi hiyerarşik yapıdadır ve seviye içermektedir. Birinci seviye de tasarım kalitesi nitelikleri tanımlanmıştır. Tanımlanan tasarım kalitesi nitelikleri; yeniden kullanılabilirlik (reusability), işlevsellik (functionality), etkinlik (effectiveness), anlaşılırlık (understandability), genişletilebilirlik (extendibility), esneklik (flexibility)' tir.

İkinci seviyede ise tasarım kalitesi özellikleri belirlenmiştir. Tasarım kalitesi özellikleri yazılımdaki tasarım birimlerinin niteliklerini, işlevlerini ve aralarındaki ilişkileri doğrudan ele almaktadır. Bu özellikler bir veya birden çok tasarım metrikleri değerlendirebilmektedir [12]. Modelde kullanılan tasarım özellikleri ise Tablo 1'de gösterilmiştir.

Üçüncü seviyede ise nesneye dayalı tasarım metrikleri tanımlanmıştır. Tasarım özelliklerinin hangi tasarım metriği ile ölçüleceğine karar verilirken QMOOD'da yer alan nesneye dayalı tasarım metrikleri incelenmiştir. İncelenen tasarım metrikleri ile iPlasma aracında yer alan metrikler karşılaştırılmış ve örtüşen metrikler seçilmiştir. Örtüşmeyen metrikler için iPlasma aracında yer alan ve tasarım özellikleriyle ilişkilendirilebilen başka metrikler seçilmiştir. Seçilen metrikler Tablo 2'de gösterilmiştir. Dördüncü seviyede ise hangi nesneye dayalı tasarım bileşenlerinin kullanılacağına karar verilmiştir. Bileşen olarak paketler, sınıflar, nitelikler, metotlar ve bunların arasındaki ilişkiler seçilmiştir. Bu bileşenler kullanılarak nesneye dayalı tasarım metrikleri elde edilebilir.

Tablo 1. QMOOD tasarım özellikleri.

1	Design Size (Tasarım Boyutu)
2	Hierarchy (Hiyerarşi)
3	Abstraction (Soyutlama)
4	Encapsulation (Kapsülleme)
5	Coupling (Bağlama)
6	Cohesion (Birleşme)
7	Composition (Kompozisyon)
8	Inheritance (Kalıtım)
9	Polymorphism (Çokşekillilik)
10	Messaging (Mesajlaşma)
11	Complexity (Karmaşıklık)

Tablo 2. Seçilen nesneye dayalı tasarım metrikleri.

Tasarım Metrikleri	Kısaltma	Açıklama
Design Class Size	DSC	Tasarımdaki toplam sınıf sayısı
Number of Hierarchies	NOH	Tasarımdaki toplam sınıf hiyerarşisi sayısı
Abstraction Ratio	AR	Bir paketteki **interface** veya **abstract** olarak tanımlanan sınıf sayısının tüm sınıfların sayısına oranı
Data Access Metric	DAM	Bir sınıftaki **private** veya **protected** olarak tanımlanan nitelikleri tüm niteliklerin sayısına oranı
Coupling Between Objects	CBO	Bir sınıfın bağlı olduğu sınıfların sayısı
Tight Class Cohesion	TCC	Bir sınıftaki doğrudan bağlı metot çiftleri sayısının tüm olası metot çifti sayısına oranı
Number Of Attributes	NOA	Bir sınıfta yer alan niteliklerin sayısı
Depth of Inheritance Tree	DIT	Bir sınıfın türetim ağacındaki köke uzaklığı.
Number of Overridden Methods	NOOM	Polimorfik davranış gösteren metot sayısı
Class Interface Size	CIS	Bir sınıfta **public** olarak tanımlanmış metot sayısı
Number of Methods	NOM	Bir sınıfta tanımlı metot sayısı

Metrikler belirlendikten sonra tasarım kalitesi özellikleri ile seçilen tasarım metrikleri ilişkilendirmiştir. Bu ilişkilendirme Tablo 3'te gösterilmiştir. Tasarım kalitesi özelliklerinin tasarım kalitesi niteliklerini hangi yönde ve ne kadar etkileyeceğine sezgisel olarak karar verilebilir. Bu çalışmada QMOOD' var olan ilişkilendirme ve ağırlıklandırma temel alınarak Tablo 4'te gösterilen ağırlıklar belirlenmiştir. Belirlenen ağırlıklar kullanılarak her bir tasarım niteliğine ait kalite puanı belirlenir. Sistemin toplam kalitesi 6 kalite niteliğine ait kalite puanın toplanmasıyla elde edilen Toplam Kalite İndeksi ile bulunur.

Tablo 3. Tasarım özellikleri-tasarım metrikleri ilişkisi.

Tasarım Metrikleri	Kısaltma
Design Size	Design Class Size (DSC)
Hierarchy	Number of Hierarchies (NOH)
Abstraction	Abstraction Ratio (AR)
Encapsulation	Data Access Metric (DAM)
Coupling	Coupling Between Objects (CBO)
Cohesion	Tight Class Cohesion (TCC)
Composition	Number of Attributes (NOA)
Inheritance	Depth of Inheritance Tree (DIT)
Polymorph	Number of Protected Methods (NOOM)
Messaging	Class Interface Size (CIS)
Complexity	Number of Methods (NOM)

Tablo 4. Tasarım kalitesi nitelikleri ile tasarım özellikleri arasındaki ilişki ve ağırlıkları (n: negatif, p: pozitif).

	Reusability	Flexibility	Understandability	Functionality	Extendibilty	Effectiveness
Design Size	p		n	p		
Hiearchy				p		
Abstraractipn			n		p	P
Encapsulation		p	p			P
Coupling	n	n	n		n	
Cohesion	p		p	p		
Composition		p				p
Inheritence					p	p
Polymorph		p	p	p	p	p
Messaging	p			p		
Complexity			p			
	Reusability	**Flexibility**	**Understandability**	**Functionality**	**Extendibilty**	**Effectiveness**
Design Size	0,5		-0,33	0,22		
Hiearchy				0,22		
Abstraractipn			-0,33		0,5	0,2
Encapsulation		0,4	0,33			0,2
Coupling	-0,25	-0,25	-0,33		-0,5	
Cohesion	0,25		0,33	0,12		
Composition		0,35				0,2
Inheritence					0,5	0,2
Polymorph		0,5	-0,33	0,22	0,5	0,2
Messaging	0,5			0,22		
Complexity			-0,33			
Toplam	1	1	-1	1	1	1

V. SONUÇLAR

Yazılımlar ile ilgili gözlemlere dayanan genel sonuçlara baktığımızda, bir yazılımın yeni sürümünün yayınlanmasıyla yazılıma yeni yetenekler ve özellikler katılabilir veya yazılımdaki hataların düzeltilmesi sağlanabilir. Bundan dolayı yazılımın çıkan ilk sürümlerinde yapısal değişiklikler fazla olur ve kalite değişimindeki büyük artışlar görülebilir. Aynı zamanda yazılımın kullanışlılığı da artabilir. İlerleyen sürümlerde ise yazılımdaki hatalar düzeltilir ve yazılımın sağlamlığı ve güvenilirliği arttırılır. Çalışmada oluşturulan modelin üretmesi beklenen sonuçlar ise aşağıdaki gibidir:

1. Kalite niteliklerinden yeniden kullanılabilirlik (reusability), işlevsellik (functionality), etkinlik (effectiveness), genişletilebilirlik (extendibility) ve esneklik (flexibility) her sürümde belli oranda artmalıdır. Bunun tersi olarak anlaşılırlık ilk sürümlerde düşmelidir [12].

2. İlk sürümlerde yazılıma fazla sayıda sınıflar ve metotlar eklendiğinden işlevsellik (functionality) ve etkinlik (effectiveness) bu sürümlerde daha fazla artmalıdır.

Weka yazılımının 3.7 sürümü geliştirici sürümü olarak bilinmektedir. Bu sürümün 12 tane alt sürümü bulunmaktadır. 12 alt sürüme ait kalite puanlarını hesaplamak için Weka'nın kaynak kodları elde edilmiş ve iPlasma metrik toplama aracı kullanılarak her bir sürüm için seçilen metrikler elde edilmiştir. Metriklerin bir kısmı sistem seviyesinde (DCS, NOH), bir kısmı ise sınıf seviyesindedir. Sınıf seviyesinde elde edilen metriklerin sistem seviyesine çıkarılması için sınıf seviyesindeki metriklerin ortalaması alınmıştır. Weka'nın 12 alt sürümüne ait metrik değerleri Tablo 5'te gösterilmiştir. Her bir metriğin aldıkları değer aralıkları farklı olduğu için kalite özelliklerinin değişimlerini yorumlamak zordur. Bu durumda sürümler arasında kalite özelliklerinin değişiminin incelenmesi gerekmektedir. Bu işlemi yapabilmek için ilk sürüm referans alınarak diğer sürümlerin metrik değerleri normalize edilmiştir. Normalize edilen bu değerler Tablo 6'da gösterilmiştir.

Tablo 5. Weka'nın 12 alt sürümüne ait metrik değerleri.

Versiyon	DSC	NOH	AR	DAM	NOA	CBO	TCC	DIT	NOOM	CIS	NOM
3.7.0	1101	466	6,44	7,13	0,803	5,428	0,222	1,332	4,640	14,585	17,268
3.7.1	1175	488	12,89	7,07	0,800	5,380	0,220	1,329	4,670	14,580	17,235
3.7.2	972	432	13,96	6,56	0,773	4,845	0,237	1,319	4,364	13,631	16,251
3.7.3	977	436	13,97	6,58	0,770	4,883	0,240	1,321	4,362	13,669	16,289
3.7.4	981	441	13,99	6,65	0,770	4,878	0,239	1,328	4,402	13,772	16,403
3.7.5	1002	451	14,02	6,67	0,766	4,888	0,235	1,334	4,409	13,772	16,384
3.7.6	1017	463	14,02	6,74	0,764	4,911	0,234	1,348	4,446	13,846	16,481
3.7.7	1022	465	14,01	6,74	0,763	4,913	0,235	1,348	4,454	13,875	16,512
3.7.8	1043	477	14,19	6,72	0,760	4,895	0,234	1,347	4,468	13,985	16,601
3.7.9	1043	477	14,19	6,73	0,760	4,897	0,234	1,347	4,468	13,985	16,601
3.7.10	1066	483	14,41	6,68	0,761	4,887	0,234	1,342	4,432	13,886	16,476
3.7.11	1310	720	14,89	6,34	0,802	4,244	0,229	1,105	3,685	13,144	15,265

Tablo 6. İlk sürüme göre normalize edilmiş metrik değerleri.

Versiyon	DSC	NOH	AR	DAM	NOA	CBO	TCC	DIT	NOOM	CIS	NOM
3.7.0	1	1	1	1	1	1	1	1	1	1	1
3.7.1	1,067	1,047	2,002	0,992	0,996	0,991	0,992	0,998	1,006	1,000	0,998
3.7.2	0,883	0,927	2,168	0,920	0,962	0,893	1,067	0,991	0,940	0,935	0,941
3.7.3	0,887	0,936	2,169	0,923	0,958	0,900	1,078	0,992	0,940	0,937	0,943
3.7.4	0,891	0,946	2,172	0,933	0,958	0,899	1,073	0,998	0,949	0,944	0,950
3.7.5	0,910	0,968	2,177	0,935	0,954	0,901	1,056	1,002	0,950	0,944	0,949
3.7.6	0,924	0,994	2,177	0,945	0,951	0,905	1,055	1,012	0,958	0,949	0,954
3.7.7	0,928	0,998	2,175	0,945	0,950	0,905	1,055	1,013	0,960	0,951	0,956
3.7.8	0,947	1,024	2,203	0,942	0,946	0,902	1,052	1,012	0,963	0,959	0,961
3.7.9	0,947	1,024	2,203	0,944	0,946	0,902	1,052	1,012	0,963	0,959	0,961
3.7.10	0,968	1,036	2,238	0,937	0,947	0,900	1,055	1,008	0,955	0,952	0,954
3.7.11	1,190	1,545	2,312	0,889	0,998	0,782	1,030	0,830	0,794	0,901	0,884

Normalize edilen metrik değerleri ve Tablo 4 yer alan ağırlıklandırılmış tasarım özellikleri kullanılarak, her sürüm için tasarım kalite niteliklerinin değeri hesaplanmıştır. Kalite niteliklerinin değerleri Tablo 7 'de gösterilmiştir. Her sürüm için kalite niteliklerinin aldığı puanlar toplanarak Toplam Kalite İndeksi hesaplanmıştır.

Tablo 7. Hesaplanan kalite niteliği değerleri.

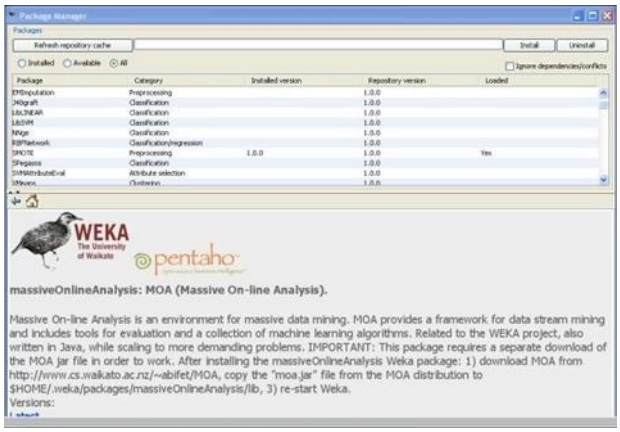
Versiyon	Reusability	Flexibility	Understandability	Functionality	Extendibility	Effectiveness	TKİ
3.7.0	1,000	1,000	-1,000	1,000	1,000	1,000	4,01
3.7.1	1,034	1,001	-1,359	1,025	1,507	1,199	4,42017709
3.7.2	0,952	0,954	-1,265	0,939	1,603	1,196	4,391632496
3.7.3	0,957	0,951	-1,268	0,943	1,601	1,197	4,393892645
3.7.4	0,961	0,959	-1,276	0,949	1,610	1,202	4,418055161
3.7.5	0,966	0,959	-1,292	0,957	1,614	1,204	4,420355675
3.7.6	0,974	0,964	-1,304	0,968	1,621	1,209	4,445122266
3.7.7	0,977	0,964	-1,307	0,971	1,621	1,209	4,448500657
3.7.8	0,991	0,964	-1,326	0,983	1,638	1,213	4,475512045
3.7.9	0,990	0,965	-1,327	0,983	1,638	1,214	4,475683493
3.7.10	0,999	0,959	-1,338	0,987	1,650	1,217	4,487580442
3.7.11	1,108	0,912	-1,311	1,098	1,577	1,165	4,561320189

VI. DEĞERLENDIRME

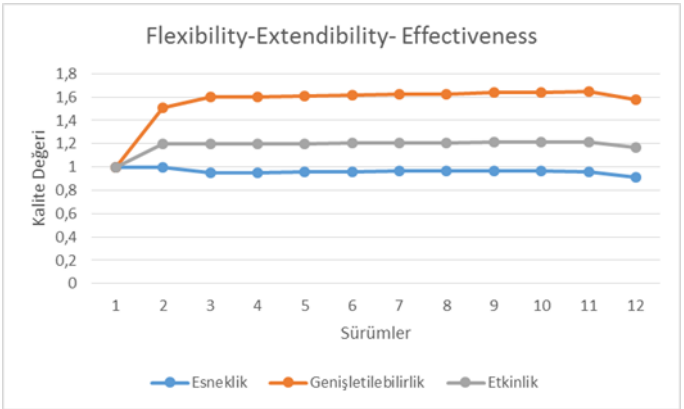
Bir önceki bölümde hesaplanan Toplam Kalite İndeksi puanlarının sürümler arası değişimleri Şekil 1'deki grafikte gösterilmiştir. Kalite niteliklerinin değerleri ayrı olarak incelendiğinde işlevsellik, esneklik ve yeniden kullanılabilirlik 3.7.0 sürümünden 3.7.1 sürümüne geçerken artmıştır. Weka'ya ait kod kayıtları (Codelogs [24]) incelendiğinde 3.7.1 sürümünde birçok sınıflandırma ve kümeleme algoritmalarının, öznitelik seçme ve veri önişleme yöntemlerinin eklendiği görülmüştür.

Şekil 1. Sürümler arası toplam kalite indeksinin değişimi.

3.7.1 sürümünden 3.7.2 sürümüne geçildiğinde ise bu üç kalite niteliğinde belirgin düşüş olmuştur. Bundaki en büyük neden Weka yazılımının tasarımında yapılan değişikliktir. Weka'nın 3.7.2 sürümünde tek monolitik çalıştırılabilir jar dosyasından, modüler paket tabanlı sisteme geçmiştir (Şekil 2). 3.7.2 sürümünden en son sürüme kadar işlevsellik ve yeniden kullanılabilirlikte belirli oranda artış olmuştur. Ancak 3.7.10 sürümünden 3.7.11 sürümüne geçerken esneklikte düşüş meydana gelmiştir. Bu duruma neden ise Weka'nın kalıtım yapısında değişiklik olmasıdır. Kod kayıtları incelendiğinde Weka'nın "core" paketinde yer alan algoritmaların hızlandırıldığı ve kodlarda geliştirilme yapıldığı görülmüştür. Aynı zaman da bazı algoritmaların hafıza tüketimlerinde azalma olmuştur (Şekil 3).

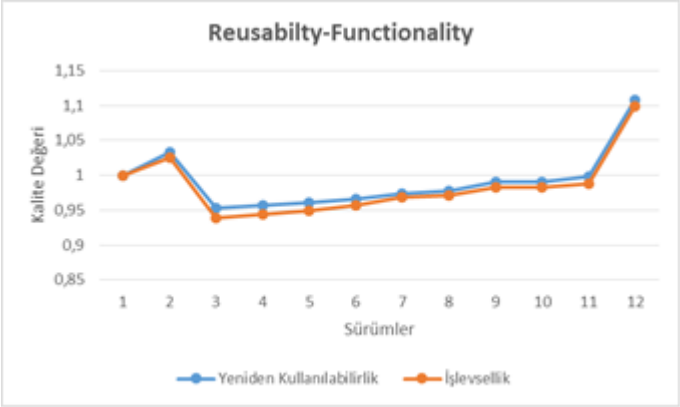


Şekil 2. Weka paket yöneticisi

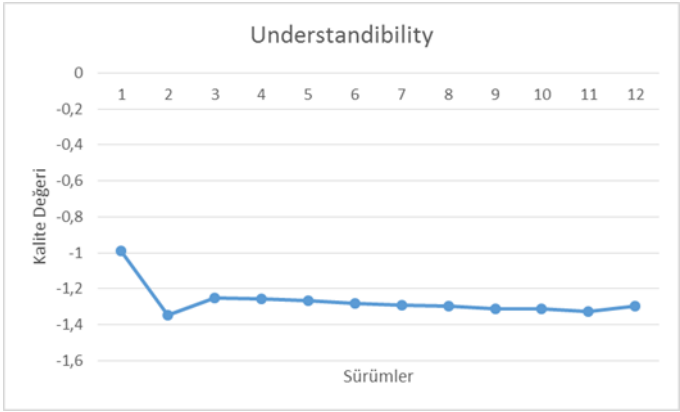


Şekil 3. Esneklik, genişletilebilirlik ve etkinlik kalite niteliklerinin sürümler arası değişimi

Etkinlik ve genişletilebilirlik nitelikleri ise benzer davranış göstermektedir. İki nitelikte de 3.7.0 sürümünden 3.7.1 sürümüne geçerken gözle görülür derecede artış görülmüştür. 3.7.1 sürümünden 3.7.10 sürümüne kadar artış devam ederken 3.7.10 sürümünden son sürüme geçişte ise düşüş meydana gelmiştir. Son sürümde kalıtım hiyerarşindeki değişiklik bu iki niteliği de etkilemiştir. Son sürümde temel sınıf sayısında belirgin derecede artış olmuştur ve 3.7.10 sürümündeki tasarım tamamen değişmiştir (Şekil 4).



Şekil 4. Yeniden kullanılabilirlik ve işlevsellik kalite niteliklerinin sürümler arası değişimi.



Şekil 5. Anlaşılırlık kalite niteliğinin sürümler arası değişimi.

Anlaşılırlık niteliğinde ise 3.7.0 sürümünden 3.7.1 sürümüne geçerken düşüş meydana gelmiştir. 3.7.1 sürümünde artan metot ve sınıf sayısı anlaşılırlığı düşürmüştür. 3.7.1 sürümünden 3.7.2 sürümüne geçildiğinde ise anlaşılırlık artmıştır. Weka'daki tasarım değişikliği bu durumda en büyük etkendir. Anlaşılırlık 3.7.3 sürümünden son sürüme kadar azalma göstermiş, son sürümde ise hafif olarak artmıştır (Şekil 5). Kurulan model ile Weka yazılımının kod kayıtları incelendiğinde tutarlılık olduğu görülmektedir. Bu durum QMOOD'un sürümler arası kalite değişimlerinin bulunmasında etkili bir model olduğunu göstermektedir.

KAYNAKLAR

[1] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, *11*(1), 10-18.

[2] Bansiya, J., & Davis, C. G. (2002). A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on software engineering*, *28*(1), 4-17.

[3] Marinescu, C., Marinescu, R., Mihancea, P. F., & Wettel, R. (2005). *iPlasma: An integrated platform for*

quality assessment of object- oriented design.

[4] Jetter, A., Gall, H., Pinzger, M., & Knab, P. (2006). Assessing software quality attributes with source code metrics.

[5] Dromey, R. G. (1995). A model for software product quality. *IEEE Transactions on software engineering*, *21*(2), 146-162.

[6] Lanza, M., & Marinescu, R. (2007). *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer Science & Business Media.

[7] Padhy, N., Satapathy, S., & Singh, R. (2017). Utility of an object oriented reusability metrics and estimation complexity. *Indian J. Sci. Technol*, *10*(3), 1-9.

[8] McCall, J. A. (1977). Factors in software quality. *US Rome Air development center reports*.

[9] Basili, V. R. (1992). *Software modeling and measurement:* The *Goal/Question/Metric paradigm*.

[10] Caldiera, V. R. B. G., & Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of software engineering*, 528-532.

[11] ISO "ISO, IEC 9126", http://www.iso.org.

[12] İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Yazılım Tasarımı Kalitesi Ders Notu. (Doç. Dr. Feza Buzluca)

[13] Jorgensen, P. C. (2018). *Software testing: a craftsman's approach*. CRC press.

[14] Gil, Y., & Lalouche, G. (2017). On the correlation between size and metric validity. *Empirical Software Engineering*, *22*(5), 2585-2611.

[15] Malhotra, R. (2016). *Empirical research in software engineering: concepts, analysis, and applications*. CRC Press.

[16] Herbold, S., Trautsch, A., & Grabowski, J. (2017). A comparative study to benchmark cross-project defect prediction approaches. *IEEE Transactions on Software Engineering*, *44*(9), 811-833.

[17] Goyal, P. K., & Joshi, G. (2014, February). QMOOD metric sets to assess quality of Java program. In *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)* (pp. 520-533). IEEE.

[18] Rathore, S. S., & Kumar, S. (2019). A study on software fault prediction techniques. *Artificial Intelligence Review*, *51*(2), 255-327.

[19] Chawla, M. K., & Chhabra, I. (2013). Capturing OO Software metrics to attain quality attributes–a case study. *International Journal of Scientific & Engineering Research*, *4*(6), 359-363.

[20] Singh, H., & Hassan, S. I. (2015). Effect of SOLID design principles on quality of software: An empirical assessment. *International Journal of Scientific and Engineering Research*, *6*(4).

[21] Gupta, M., & Singh, S. (2018). Comparative Analysis of Software Design Patterns Based Design Metrics using Machine Learning Algorithms. *Journal of Computer Engineering & Technology*, *9*(3), 32-41.

[22] Radjenović, D., Heričko, M., Torkar, R., & Živkovič, A. (2013). Software fault prediction metrics: A systematic literature review. *Information and software technology*, *55*(8), 1397-1418.

[23] URL: www.cs.waikato.ac.nz/ml/weka/, (Erişilme Tarihi: 20.11.2019)

[22] URL:http://wiki.pentaho.com/display/DATAMINING/Pentaho+Data+Mining+Community+ Documentation*, (Erişilme Tarihi:20.11.2019)