# Android Platformu için Hazırlanan Mobil Uygulamanın Kalite Nitelikleri Değişiminin İncelenmesi

Yusuf Özçevik, 504152517
Bilgisayar Mühendisliği Bölümü
İstanbul Teknik Üniversitesi
İstanbul, Türkiye
ozcevik@itu.edu.tr

***Özetçe* —Bu çalışmada Java dili ile Android platformu için hazırlanmış bir mobil uygulamanın bir yıllık süre zarfı içerisindeki ilk 9 versiyonu incelenmiştir. İncelenen versiyonlar için Chidember-Kemerer Metrik Kümesi ve QMOOD Modeli kullanılarak tasarım kalitesi nitelikleriyle ilgili sayısal ölçümler elde edilmiştir. Uygulamada yapılan versiyon güncellemeleriyle birlikte ilgili niteliklerin değişimi incelenerek elde edilen sonuçlar değerlendirilmiştir.**

***Anahtar Kelimeler—Tasarım Kalitesi Nitelikleri, Nesneye Dayalı Programlama, QMOOD***

***Abstract*—In this work, first nine versions of a mobile application for Android platform which is implemented by Java programming language is studied. Chidember-Kemerer Metric Set and QMOOD Model is used for the investigation between application versions. Considering different versions of the application, the change of computed quality attributes are evaluated.**

***Keywords—Quality Attributes, Object Oriented Programming, QMOOD***

## I. GİRİŞ

Son yıllarda yazılım teknolojilerinin farklı endüstriyel alanlarda farklı işlevlere yönelik bir çok cihazda yaygın olarak kullanılmaya başlanması, kullanılan yazılımların değerlendirilerek geri besleme yapılması ihtiyacını doğurmuştur. Bu değerlendirme ve geri besleme ihtiyacı müşteri açısından daha kaliteli hizmet almak adına, geliştirici açısından ise müşteri memnuniyeti adına önem teşkil etmektedir. Yazılım teknolojilerinin bu denli yaygınlaşması ve yazılım paydaşlarının kalite konusundaki bilgi ihtiyacı yazılımların kalite değerlendirmesini gerekli kılar.

Varlıklar arasındaki ilişkileri belirlemek adına gözlem sıklıkla kullanılan bir yaklaşımdır. Ancak yazılım kalitesinin gözlemlenerek, aynı işlevi yerine getiren birden fazla yazılımdan hangisinin daha iyi olduğunu söylemek mümkün değildir. Yazılım kalitesinin ölçülmesinde gerçek dünyada gözlem yapmayı olanaksız kılan bilgi engelinden dolayı bu tür karşılaştırmalar ve yorumlar ölçme yoluyla dolaylı olarak yapılır. Değerlendirme gözlem yoluyla gerçek dünyada değil, ölçme ile elde edilen sayısal değerler üzerinden yapılarak yorumlanabilir.

Yazılım kalitesinin ölçülmesi ve birden fazla yazılımın kalite sonuçlarının karşılaştırılması için sezgisel (gerçek) dünya ile formel (matematiksel) dünya arasında modellemeye ihtiyaç duyulur. Bu modelleme sayesinde yazılımlar ile ilgili sayısal veriler elde edilerek kıyaslama yapılabilir. Ancak, yapılan kıyaslamanın doğru olabilmesi için kullanılan modelin sezgisel dünya varlıklarını formel dünyada doğru temsil etmesi gerekmektedir. Bu bağlamda elde edilen ölçme sonuçları sezgisel dünyadan elde edilebilen gözlem sonuçlarıyla geri beslenerek modelin doğruluğunu sınamak önem teşkil eder. Chidember-Kemerer Metrik Kümesinden [1] yararlanarak ve nesneye dayalı tasarımlar için sezgisel dünya ile formel dünya arasında bir model sunan QMOOD Modeli, temsilin doğruluğu açısından başaryla sınanmış modellerden biridir [2]. Bu bağlamda bu çalışmada QMOOD Modeli kullanılarak Android Mobil platformu için Java dili kullanılarak hazırlanmış bir mobil uygulamanın versiyonları arasındaki kalite niteliklerinin değişimi incelenmiştir. İlgili yazılıma ait bir yıllık süreçte geliştirilen ilk 9 versiyon göz önüne alınmış ve elde edilen sonuçlar değerlendirilerek QMOOD Modelinin mobil platformlar için hazırlanan nesneye dayalı mobil uygulamalar için beklenen sonuçları verip vermediği sınanmıştır.

Çalışmanın diğer kısımları şu şekilde düzenlenmiştir. Bölüm II'de konuyla ilgili literatür araştırmasına yer verilmiştir. Bölüm III'de bu çalışmada kullanılan yaklaşım, hangi kalite niteliklerinin hangi metrikler ile ölçüldüğü ve ölçüm yapmak için kullanılan araçlar belirtilmiştir. Bölüm IV'de ölçme sonuçlarının sayısal analizi yapılarak elde edilen sonuçlar grafikler aracılığıyla görselleştirilmiştir. Son olarak, Bölüm V'de elde edilen sonuçlar değerlendirilerek çalışma sonlandırılmıştır.

## II. İLGİLİ ÇALIŞMALAR

Yazılım kalitesinin ölçülmesi konusunda son yıllarda literatürde bir çok çalışma yer almaktadır. [3], [4], [5] ve [6]'da sezgisel dünya ile formel dünya arasındaki temsil için oluşturulan modeller incelenmiştir. Benzer şekilde [2]'de nesneye dayalı tasarım kalitesini değerlendirmek amacıyla oluşturulan hiyerarşik model ele alınmıştır. [7]'deki çalışmada yer alan ve McCall tarafından oluşturulan yazılım modeli, yazılım paydaşları arasında iletişim sağlamayı amaçlamıştır ve pek çok yazılıma temel oluşturmaktadır. [8] ve [9]'da Victor Basili tarafından oluşturulan Hedef/Soru/Metrik (GQM) yaklaşımı ise ilk olarak projelerde meydana gelen hataları belirlemek için oluşturulan, sonrasında ise yazılım kalitesini geliştirmeyi amaçlayan yöntemleri ele almaktadır. [10]'de ise

ISO/IEC kuruluşu, yazılım kalitesi ile ilgili tanımlamaların standartlaştırılmasını amaçlamaktadır ve bununla ilgili tanımlamalardan oluşan 4 bölümlük bir döküman ele almıştır. Bu dokümanda yazılım kalitesi ile ilgili bileşenler açık ve net olarak tanımlanmış ve bu sayede yazılım kalitesi konusunda çalışan bireyler arasında ortak bir dil oluşturulmuştur.

## III. KULLANILAN YAKLAŞIM VE ARAÇLAR

### A. Kullanılan Yaklaşım

Bu çalışmada, kullanılan projenin incelenmesi sırasında Chidember-Kemerer Metrik Kümesi ve QMOOD Modeli'nden yararlanılmıştır. QMOOD Modeli'nde proje kalitesinin ölçülmesi için dört katmandan oluşan hiyerarşik ve katmanlı bir yapı oluşturulmuştur [2]. Tanıma göre en üst katmanda kalite nitelikleri, onun bir altında tasarım özellikleri, onun bir altında tasarım metrikleri ve en altta da tasarım bileşenleri yer alır. Proje kalitesi belirlenirken katmanlar arasında yukarıdan aşağıya doğru gidilir. En alt katmanda yer alan tasarım bileşenlerinin matematiksel olarak ölçülmesinden sonra ters yönde gidilerek toplam kalite elde edilebilir. Bu bağlamda en üst katmanda belirlenen altı kalite niteliğinin hesaplanması için bir alt katmanda yer alan tasarım özellikleri kullanılır. Bu iki katman arasındaki geçiş (1), (2), (3), (4), (5) ve (6) ile formülize edilmiştir.

$$\textit{Tekrar Kullanılabilirlik}= \begin{cases} +0,5 \cdot (Desing\ Size) \\ -0,25 \cdot (Coupling) \\ +0,25 \cdot (Cohesion) \\ +0,5 \cdot (Messaging) \end{cases} \quad (1)$$

$$\textit{Esneklik}= \begin{cases} +0,25 \cdot (Encapsulation) \\ -0,25 \cdot (Coupling) \\ +0,5 \cdot (Composition) \\ +0,5 \cdot (Polymorhism) \end{cases} \quad (2)$$

$$\textit{Anlaşılırlık}= \begin{cases} -0,33 \cdot (DesignSize) \\ -0,33 \cdot (Abstraction) \\ +0,33 \cdot (Encapsulation) \\ -0,33 \cdot (Coupling) \\ +0,33 \cdot (Cohesion) \\ -0,33 \cdot (Polymorhism) \\ -0,33 \cdot (Complexity) \end{cases} \quad (3)$$

$$\textit{İşlevsellik}= \begin{cases} +0,22 \cdot (Design\ Size) \\ +0,22 \cdot (Hierachies) \\ +0,12 \cdot (Cohesion) \\ +0,22 \cdot (Polymorhism) \\ +0,22 \cdot (Messaging) \end{cases} \quad (4)$$

$$\textit{Genişletilebilirlik}= \begin{cases} +0,5 \cdot (Abstraction) \\ -0,5 \cdot (Coupling) \\ +0,5 \cdot (Inheritance) \\ +0,5 \cdot (Polymorhism) \end{cases} \quad (5)$$

$$\textit{Etkinlik}= \begin{cases} +0,2 \cdot (Abstraction) \\ +0,2 \cdot (Encapsulation) \\ +0,2 \cdot (Composition) \\ +0,2 \cdot (Inheritance) \\ +0,2 \cdot (Polymorphism) \end{cases} \quad (6)$$

Tasarım özelliklerinden bir alt katmana inmek için tasarım özelliklerini tasarım metrikleri şeklinde ifade etmek gerekecektir. Tasarım özellikleri ile tasarım metrikleri arasındaki ilişki Tablo I ile verilmiştir.

| Tasarım Özelliği | İlgili Tasarım Metriği |
|---|---|
| Design Size | Design Size in Classes |
| Hierarchies | Number of Hierarchies |
| Abstraction | Average Number of Ancestors |
| Encapsulation | Data Access Metric |
| Coupling | Direct Class Coupling |
| Cohesion | Cohesion Among Methods in Class |
| Composition | Measure of Aggregation |
| Inheritance | Measure of Functional Abstraction |
| Polymorphism | Number of Polymorphic Methods |
| Messaging | Class Interface Size |
| Complexity | Number of Methods |

Tablo I: Tasarım Metrikleri ile Tasarım Özellikleri Arasındaki İlişki

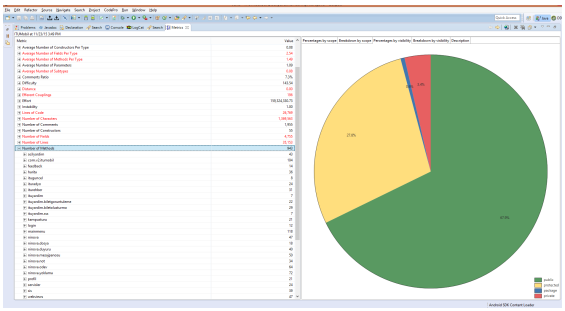
İlgili tasarım metriklerinin tasarım bileşenleri ile ifade edilerek modelde en alt katmana inilmesi ve gerekli verilerin projeden ölçülmesi için aşağıdaki tanımlar kullanılmıştır. Bu tanımlar yapılırken çalışmada kullanılan iki metrik toplama aracından alınabilen değerler göz önüne alınmıştır.

- **Design Size in Classes (DSC):** Projedeki toplam sınıf sayısı olarak değerlendirilmiştir.
- **Number of Hierarchies (NOH):** Projedeki sınıf hiyararşi sayısı olarak ele alınmıştır.
- **Average Number of Ancestors (ANA):** Projedeki kalıtım ağacı boyunun ortalaması olarak hesaplanmıştır.
- **Data Access Metric (DAM):** Projedeki sınıflarda yer alan private ve protected niteliklerin toplamının, projede yer alan tüm niteliklere oranı olarak hesaplanmıştır.
- **Direct Class Coupling (DCC):** Proejedeki sınıfların birbirleriyle bağımlılık oluşturduğu sınıf sayısı olarak ele alınmıştır.
- **Cohesion Among Methods in Class (CAM):** Metrik toplama aracından alının LCOM (Lack of Cohesion) metriği kullanılarak 1-LCOM olarak hesaplanmıştır.
- **Measure of Aggregation (MOA):** Projede yer alan kullanıcı tanımlı sınıf nesnesi sayısı olarak ele alınmıştır.
- **Measure of Functional Abstraction (MFA):** Projede yer alan bir sınıfın atasından türettiği metod sayısının toplam metod sayısına oranı olarak ele alınmıştır.
- **Number of Polymorphic Methods (NOP):** Projede yer alan interface sayısı olarak ele alınmıştır.

- **Class Interface Size (CIS):** Projedeki public metod sayısı olarak değerlendirilmiştir.
- **Number of Methods (NOM):** Projedeki toplam metod sayısı olarak ele alınmıştır.

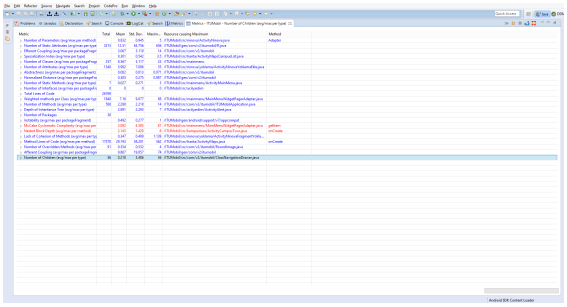
### *B. Kullanılan Araçlar*

Çalışmada incelenen mobil uygulamanın geliştirme ortamı Eclipse IDE'dir. Bu bağlamda, tasarım metrikleriyle ilişkili tasarım bileşenlerini ölçmek için iki ayrı Eclipse eklentisi kullanılmıştır. Bunlardan biri Şekil 1'de görülen CodePro Analytics [11] eklentisidir. Şekilde görüldüğü gibi bu eklenti ile elde edilen metriklerin grafiklerle görselleştirilmesi ve belirlenen eşik değerinden büyük metrik değerlerinin renklendirilerilmesi mümkündür. Bu eklenti ile projeden NOH, DAM, DCC, MOA, CIS ve NOM metrikleri ölçülmüştür.



Şekil 1: Code Pro Analytics Eklentisi ile Metrik Çıkarımı

Uygulamada kullanılan bir diğer Eclipse eklentisi açık kaynak kodlu Eclipse Metrics [12] isimli eklentidir. Eklentiden elde edilebilen metrikler Şekil 2'de gösterilmiştir. İlk eklentide bulunan görselleştirme ve renklendirme özellikleri bu eklentide de mevcuttur. Bu çalışma kapsamında projeden DSC, ANA, CAM, MFA ve NOP metrikleri bu eklenti ile elde edilmiştir.



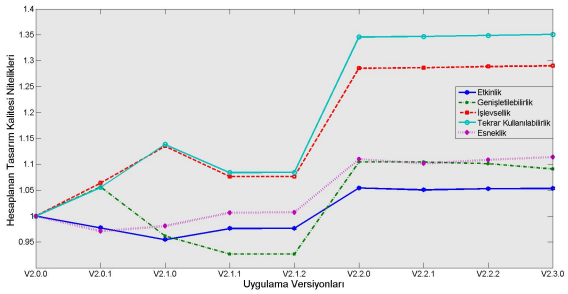
Şekil 2: Eclips Metrics Eklentisi ile Metrik Çıkarımı

## IV. DEĞERLENDİRME

Uygulamadan elde edilen tasarım bileşenleri Bölüm III'de anlatılan yaklaşım ve araçlar kullanılarak çıkarılmıştır. Bu tasarım bileşenleri ve ilgili bileşene ait sayısal değerler Tablo II ile verilmiştir.

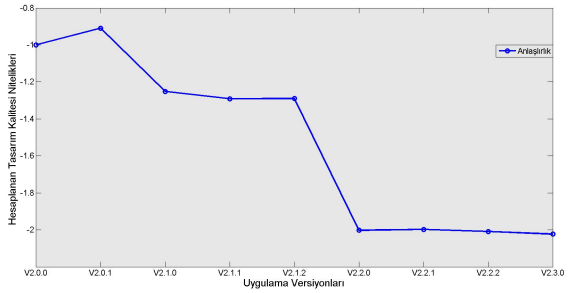
Projeden elde edilen ve Tablo II ile gösterilen tasarım bileşenleri uygun şekilde aynı düzeye çekildikten sonra normalize edilerek modelde tanımlanan hiyerarşide üst katmanlara çıkılmıştır. En üst katmana geçerken (1), (2), (3), (4), (5) ve (6) eşitlikleri kullanılarak 9 farklı proje versiyonu için kalite niteliklerine ulaşılmıştır. Bu kalite nitelikleri ilk versiyon olan V2.0.0'a göre normalize edilerek Tablo III ile gösterilmiştir.

Tablo III ile gösterilen kalite niteliklerinin farklı proje versiyonları için normalize edilmiş sayısal değerleri Şekil 3 ve Şekil 4 ile görselize edilmiştir.




Şekil 3: Farklı Uygulama Versiyonları için Etkinlik, Genişletilebilirlik, İşlevsellik, Tekrar Kullanılabilirlik ve Esneklik Niteliklerinin Değişimi

Şekil 3'de etkinlik, genişletilebilirlik, işlevsellik, tekrar kullanılabilirlik ve esneklik niteliklerinin 9 farklı proje versiyonuna göre değişimi görülmektedir. Bu niteliklere ait hesaplanan sayısal değerler y-ekseninde yer alırken; versiyon numaraları x-eksini ile verilmiştir. Sezgisel olarak, bu niteliklerin sürümler arasında artış göstermesi beklenmektedir. Formel olarak analiz edilen bu beş nitelik, özellikle projede temel geliştirmelerin yapıldığı V2.0.0-V2.1.0-V2.2.0-V2.3.0 sürüm geçişlerinde beklendiği gibi gözle görülür bir artış gösterirken; hata düzeltme ve ufak görsel düzenlemelerin yer aldığı V2.0.1-V2.1.1-.V2.1.2-V2.2.1-V2.2.2 sürüm geçişlerinde beklenen karakteristikten sapmalar göstermiştir. Bu durumun temel nedeni karşılaşılan hata düzeltme aşamalarında nesneye dayalı yazılım niteliklerinin göz önüne alınmaksızın hızlı bir şekilde güncelleme ihtiyacını gidermeye yönelik geliştirme yapılması olabilir.




Şekil 4: Farklı Uygulama Versiyonları için Anlaşılabilirlik Niteliğinin Değişimi

Şekil 4'de anlaşılabilirlik niteliğinin 9 farklı proje versiyonuna göre değişimi görülmektedir. Bu niteliğe ait hesaplanan sayısal değerler y-ekseninde yer alırken; versiyon numaraları yine x-eksini ile verilmiştir. Sezgisel olarak, anlaşılabilirlik niteliğin ilk versiyonlarda azalma göstermesi beklenir; ancak bir yerden sonra artması beklenmektedir. Grafiğe

| Tasarım Özellikleri | Mobil Uygulama Versiyonları | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | V2.2.0 | V2.0.1 | V2.1.0 | V2.1.1 | V2.1.2 | V2.2.0 | V2.2.1 | V2.2.2 | V2.3.0 |
| **Design Size** | 111 | 134 | 166 | 144 | 144 | 238 | 238 | 239 | 240 |
| **Hierarchies** | 3,090 | 3,000 | 3,050 | 3,120 | 3,120 | 2,860 | 2,860 | 2,870 | 2,870 |
| **Cohesion** | 0,701 | 0,767 | 0,731 | 0,689 | 0,690 | 0,667 | 0,671 | 0,673 | 0,670 |
| **Polymorphism** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Messaging** | 310 | 307 | 370 | 370 | 370 | 580 | 580 | 580 | 583 |
| **Abstraction** | 3,495 | 3,067 | 3,181 | 3,514 | 3,514 | 3,038 | 3,038 | 3,050 | 3,046 |
| **Encapsulation** | 0,734 | 0,730 | 0,690 | 0,696 | 0,695 | 0,652 | 0,654 | 0,652 | 0,651 |
| **Composition** | 110 | 114 | 161 | 162 | 162 | 335 | 329 | 334 | 339 |
| **Inheritance** | 0,084 | 0,084 | 0,075 | 0,075 | 0,075 | 0,095 | 0,095 | 0,094 | 0,094 |
| **Coupling** | 95 | 95 | 121 | 121 | 121 | 183 | 183 | 84 | 185 |
| **Complexity** | 440 | 437 | 540 | 540 | 540 | 850 | 850 | 854 | 860 |

Tablo II: Farklı Uygulama Versiyonları için Ölçülen Tasarım Bileşenlerinin Değerleri

| Kalite Nitelikleri | Mobil Uuygulama Versiyonları | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | V2.0.0 | V2.0.1 | V2.1.0 | V2.1.1 | V2.1.2 | V2.2.0 | V2.2.1 | V2.2.2 | V2.3.0 |
| **Reusability** | 1 | 1,055 | 1,138 | 1,084 | 1,084 | 1,345 | 1,346 | 1,348 | 1,350 |
| **Flexibility** | 1 | 0,970 | 0,980 | 1,006 | 1,007 | 1,109 | 1,101 | 1,108 | 1,114 |
| **Understandibility** | -1 | -0,909 | -1,251 | -1,290 | -1,290 | -2,002 | -1,997 | -2,009 | -2,023 |
| **Functionality** | 1 | 1.063 | 1,134 | 1,076 | 1,076 | 1,285 | 1,286 | 1,288 | 1,290 |
| **Extendibility** | 1 | 1,056 | 0,961 | 0,926 | 0,926 | 1,104 | 1,104 | 1,101 | 1,09 |
| **Effectiveness** | 1 | 0,977 | 0,954 | 0,976 | 0.976 | 1,054 | 1,050 | 1,052 | 1,053 |

Tablo III: Farklı Uygulama Versiyonları için Hesaplanan Kalite Nitelikleri

göre formel olarak analiz edilen anlaşılabilirlik niteliğinin, esas geliştirmelerin yapıldığı versiyon geçişlerindeki sayısal sıçramaları daha fazla iken ara versionlarda sabit kaldığı söylenebilir. V2.2.0-V2.3.0 aralığında anlaşılabilirlik niteliğindeki azalma hızının kırıldığı ve ilerleyen versiyonlarda artırılması adına yapılacak düzenlemelerden sonra beklenen davranışa geleceği öngörülebilir.

## V. SONUÇ

Bu çalışmada Anroid platformu için Java dili kullanılarak hazırlanan bir mobil uygulama ele alınmıştır. Bu uygulamanın bir yıl içerisinde geliştirilen ilk 9 versiyonu arasındaki kalite niteliklerinin değişimi QMOOD Modeli kullanılarak incelenmiştir. Sonuç olarak, kullanılan modelin, ilgili mobil uygulama için beklendiği gibi sonuçlar ürettiği gözlemlenerek; modelin nesneye dayalı tasarım kriterleri göz önüne alınarak hazırlanan mobil uygulamalar için doğruluğu görülmüştür. Ayrıca, kalite niteliklerinin değişim karakteristiğine bakıldığında, projenin geliştirilmesi esnasında, hata düzeltme için yapılan sürüm güncellemelerinde nesneye dayalı programlama prensiplerinden uzaklaşıldığı ya da dikkate alınmadığı savunulabilir. Esas geliştirmelerin yapıldığı sürüm geçişlerinde ise kalite niteliklerinin değişim karakteristiği beklendiği gibi olmuştur. Anlaşılabilirlik niteliğinin kalite karakteristiğinin kabul edilebilir olması adına bundan sonraki sürümlerde bu nitelikle ilgili nesneye dayalı prensiplere biraz daha özen göstermek gerektiği öz eleştirisi yapılabilir.

## KAYNAKÇA


[1] S. Chidamber and C. Kemerer, “A metrics suite for object oriented design,” IEEETransactions on Software Engineering, vol. 20, No. 6, pp. 476-493, 1994.

[2] Bansiya, J. and Davis, C. G., "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, vol. 28, no. 1, pp.4-17, (2002).

[3] Jetter, A., “Assessing Software Quality Attributes With Source Code Metrics”, Diploma Thesis, University of Zurich Department of Informatics, Zurih, October (2006).

[4] Dromey, R. G., “A Model For Software Product Quality”, Software Engineering, IEEE Transactions on, 21(2):146-162 (1995).

[5] M. Lanza and R. Marinescu, Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer, Heidelberg (2006).

[6] Rosenberg, L.,H., “Applying And Interpreting Object Oriented Metrics”, Proc. Software Technology Conference. Utah, (1998)

[7] McCall, J. A., Richards, P. K. and Walters, G. F., "Factors in Software Quality", Nat’l Tech.Information Service, no. Vol. 1, 2 and 3, (1977).

[8] Victor R. Basili, “Software Modeling and Measurement: The Goal/Question/Metric Paradigm”, Instute for Advanced Computer Studies. Department of Computer Science, University of Maryland, 1992.

[9] Victor R. Basili, Gianluigi Caldiera, H. Dieter Rombach, “The Goal Question Metric Approach”, Encyclopedia of Software Engineering, Wiley1994.

[10] ISO “ISO,IEC 9126”,http://www.iso.org

[11] Google Developers, ’Codepro AnalytiX | Google Developers’, 2015. [Online]. Available: https://developers.google.com/java-dev-tools/codepro/

[12] Metrics.sourceforge.net, ’Metrics 1.3.6’, 2015. [Online]. Available: http://metrics.sourceforge.net/.