Pulumi Bulut Yazılımı Oluşturma Aracı

Seher Aydın,162805011
Yazılım Mühendisliği Bölümü
Celal Bayar Üniversitesi
Manisa, Türkiye

***Özetçe* —Bu çalışmada C# dili ile hazırlanmış bir bulut yazılımı oluşturma aracının 9 versiyonu incelenmiştir. Açık kaynak kodlu bir yazılımdır. Gelişimi hala devam etmektedir. İncelenen versiyonlar için Chidember-Kemerer MetricSet, diğer metrikler ile QMOOD Modeli kullanılarak tasarım kalitesi nitelikleriyle ilgili sayısal ölçümler elde edilmiştir. Uygulamada yapılan versiyon güncellenmeleriyle ilgili nitreliklerin değişimi incelenilerek elde edilen sonuçlar değerlendirilmiştir.**

Anahtar Kelimeler—Nesneye Dayalı Programlama, QMOOD, OOP, Tasarım Kalitesi Nitelikleri, NDepended

***Abstract*—In this work, nine versions of a application is implemented by C programming language is studied. Chidember-Kemerer MetricSet 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

Yazılım teknolojilerinin farklı alanlarda farklı görevlerde teknlojiye yönelik bir çok cihazda yaygın olarak kullanılmaya başlanması, kullanılan yazılımların değerlendirilerek geri besleme yapılması ihtiyacını doğurmuştur. 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 karşılaştırmalar yapmak ve yorumlar ölçme yoluyla dolaylı olarak yapılmaktadır. Değerlendirme gözlem yoluyla gerçek dünyada degil, ölçme ile elde edilen sayısal veri üzerinden yapılarak yorumlanabilir. Chidember-Kemerer Metrik Kümesinden [1].
yararlanarak ve nesneye dayalı tasarımlar için sezgisel dünya ile formel dünyaarasında bir model sunan QMOOD Modeli, temsilin doğruluğu açısından başarıyla sınanmıs modellerden biridir [2]. Bu bağlamda, bu çalışmada QMOOD Modeli kullanılarak Pulumi adlı yazılım, windows platform için C programala dili kullanılarak hazırlanan bu uygulamanın V.1.0.0-V1.4.0 versiyon aralıklarını kalite niteliklerinin değişimi açısından incelenmiştir. Bu uygulama açık kaynak kodludur. Gelişimi hala devam etmektedir.

II. İNCELENEN YAZILIM

Pulumi'nin Kod SDK Olarak Altyapısı , herhangi bir bulut üzerinde kapsayıcılar, sunucusuz işlevler, barındırılan hizmetler ve altyapı kullanan bulut yazılımı oluşturmanın ve dağıtmanın en kolay yoludur [3].Biz bu sebeplerden dolayı pulumi inceledik bu yazılımın incelemesinde Visual Studio Community 16.8.3 ve ndepend Version v2020.2.1 araçlarını kullandık.

III. İLGİLİ ÇALIŞMALAR

Yazılım kalitesinin ölçülmesi konusunda son yıllarda literatürde bir çok çalısma yer almaktadır [5], [6], [7], [8]'da sezgisel dünya ile formel dünya arasındaki temsil için olusturulan modeller incelenmiştir .Benzer şekilde [2]'de nesneye dayalı tasarım kalitesini degerlendirmek amacıyla oluşturulan hiyerarşik model ele alınmıştır. [9]'daki çalısmada yer alan ve McCall tarafından oluşturulan yazılım modeli, yazılım paydaşları arasında iletişim sağlamayı amaçlamıştır ve pek çok yazılıma temel oluşturmaktadır.

IV. KULLANILAN YAKLAŞIM VE ARAÇLAR

QMOOD bir yazılım kalitesi inceleme modelidir. Modeller, bir yazılımın farklı parametrelerini degerlendirip sonucunda insan için anlamlı çıktılara ulaşmaya çalışırlar.Bu çıkarımlara ulaşmak adına QMOOD, yazılımın ham verilerininin(L4) yanısıra yazılım nitelikleri(L1), yazılım özellikleri(L2), yazılım metrikleri(L3) olarak düzeylendirilmiştir.QMOOD modeli temelinde bir yazılımın kalitesini 11 temel metrik ve bunların etkiledigi gözlemlenebilir 6 nitelikle inceler.QMOOD kapsamında kullanıcıya en çok hitap eden parçalar yazılım nitelikleridir. Bu nitelikler modelin daha küçük parçalardan ulaştıgı sonuçlardır Bu bölümde, kullanılan yaklaşım ve kullanılan araçlara yer vericeğiz.

A. Kullanılan Yaklaşım

Şu şekilde formülize edilirler (L1 - L2 Geçicisi) :

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

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

$$Anlaşırılık = \begin{cases} -0,33 \cdot (DesingSize) \\ -0,33 \cdot (Abstraction) \\ +0,33 \cdot (Encapsulation) \\ -0,33 \cdot (Coupling) \\ +0,33 \cdot (Cohesion) \\ -0,33 \cdot (Polymorphsim) \\ +0,33 \cdot (Complexity) \end{cases} \quad (3)$$

$$İşlevısellik = \begin{cases} +0,22 \cdot (DesignSize) \\ +0,22 \cdot (Hierarchies)) \\ +0,12 \cdot (Cohesion) \\ +0,22 \cdot (Polymorphsim) \\ +0,12 \cdot (messaging) \end{cases} \quad (4)$$

$$Genileti\imath ebilirlik = \begin{cases} +0,5 \cdot (Abstraction) \\ -0,5 \cdot (Coupling) \\ +0,5 \cdot (Inheritance)) \\ +0,5 \cdot (Polymorphism) \end{cases} \quad (5)$$

$$Etkinılık = \begin{cases} +0,20 \cdot (Abstraction) \\ +0,20 \cdot (Encapsulation)) \\ +0,20 \cdot (Composition) \\ +0,20 \cdot (Inheritance) \\ +0,20 \cdot (Polymorphism) \end{cases} \quad (6)$$

Yazılım nitelikleri ile yazılım özellikleri arasındaki ilişki,birçok özelliğin harmanlayıp bizlere daha özütülmüş bilgi vermeyi amaçlamaktadır Bir alt seviyeye inildiginde diger bir düzey olan yazılım metrikleri bizi karsılamaktadır. Yazılım metrikleri, temel anlamda kodun daha somut ve sayılabilir verilerini (Satır Sayısı,Sınıf Sayısı, Kod satırı Sayısı gibi.) matematiksel hesaplamalar veya gösterimler adına daha kolay işlenebilir hale getirmektedir.

Tasarım Özelliği	İlgili Tasarım Metriği
Design Size	Design Size in Classes
Complexity	Number of Methods
Hierarchies	Number Hierarchies (NOH)
Messaging	Class Interface Size (CIS)
Composition	Measure Aggregation (MOA)
Encapsulation	Data Access Metric (DAM)
Coupling	Direct Class Coupling (DCC)
Inheritance	Measures of Functional Abstraction (MFA)
Polymorphism	Number of Polymorphic Methods (NOP)
Cohesion	Cohesion Among Methods (CAM)
Abstraction	Average Number of Ancestors (ANA)

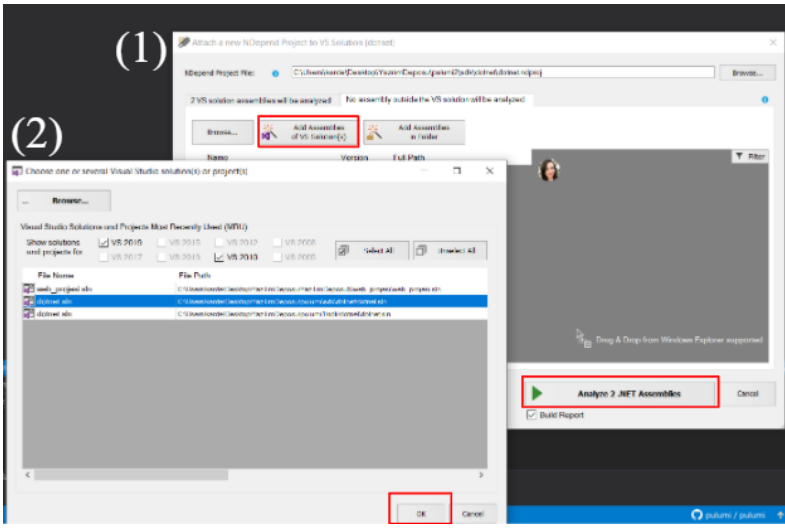
Tablo I: Tasarım Metrikleri ile Tasarım Özellikleri Arasındaki İlişki

Örnek bir item list:

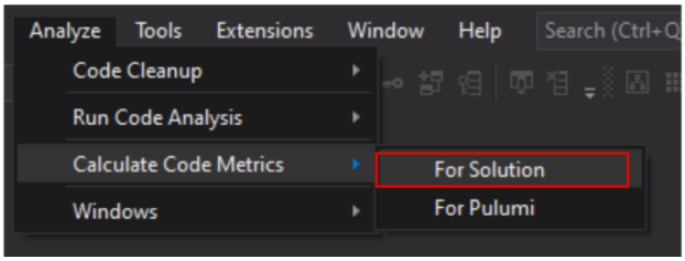
- **Design Size in Classes (DSC):** Projedeki toplam sınıf sayısı olarak değerlendirilmiştir.
- **Number of Hierarchies (NOH):** Projedeki sınıf hiyararşi sayısı olarak ele alınmıştır.
- **Average Number of Ancestors (ANA):** Projedeki kalıtım ağacı boyunun ortalaması olarak hesaplanmıştır.
- **Class Interface Size (CIS):** Sınıfın açık (public) metotlarının sayısıdır ve yazılımın mesajlaşma özelliği hakkında fikir verir.
- **Measure Of Aggregation (MOA) :** Kullanıcı tarafından tanımlanmı¸s sınıfların temel sistem veri tiplerine (int, char, double vs. . .) oranıdır.
- **Data Access Metric (DAM) :** : Sınıfın özel(private) ve korumalı(protected) niteliklerinin tüm niteliklere oranıdır.
- **Cohesion Among Methods (CAM) :** Metotların imzaları arasındaki benzerligin ölçüsüdür.
- **Direct Class Coupling (DCC) :** : Birbiriyle doğrudan ilişkili olan sınıfların sayısıdır.
- **Inheritance (MFA):** : Sınıfın kalıtımla aldıgı metot sayısının kendi barındırdığı metot sayısı oranıdır.
- **Number of Polymorphism (NOP) :** Kalıtılan toplam metot sayısıdır.
- • **Number of Methods (NOM) :** Sınıfın metot sayısıdır.

B. Kullanılan Araçlar

• **Ndepend ve Visual Studio Community :** Bir visual Studio eklentisi olan ndepend ile c# projeleri metrikleri teste tabii tutulabilmektedir. [4]. Visual Studio Community ise Visual Studio içerisinden halii hazırda erişilebilir bir araçtır.



Şekil 1: Ndepend Eklentisi ile Metrik Çıkarımı



Şekil 2: Visual Studio Community Eklentisi ile Metrik Çıkarımı

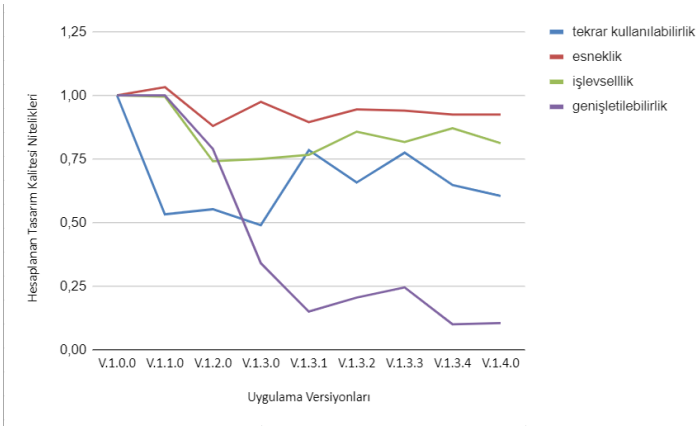
V. DEĞERLENDİRME

A. *Tasarım Niteliklerinin Bulunması*

İncelediğimiz projenin tasarım niteliklerini yalın bir şekilde açıklayabilmek için niteliklerin sınıflara göre ağırlıklı ortalamasını alıp daha sonrasında bu değerleri belli bir sayı aralığına indirgemek için de bu niteliklerin standart sapmalarını hesaplayıp sonrasında grafiğe aktarma yoluna gittik.

Yukarıda resimlerde gördüğünüz üzere ndepend eklentisi ile visual studio içerisinde hem visual studio'nun vermiş olduğu eklentiler kullanılarak hemde ndepend eklentisi aracılığı ile açık kaynak kodlu yazılımın metrik değerleri çıkarılabilmektedir. Grafikte kullandığımız değerler aşağıda verilmiştir;

Projede bulunan 9 farklı versiyon için farklı metrik değerleri gerek hem visual studio gerekse ndepend eklentisi aracılığı ile teste tabii tuttuğumuzda çıkan sonuçları normalizasyon işleminden geçirdikten sonra exel aracılığa ile grafiğe döktüğümüz zaman elimizde aşağıda gözüken tarzda tablolar ortaya çıkmıştır.



Şekil 3: Farklı Versiyonlar için Etkinlik, Genişletilebilirlik, İşlevsellik,Tekrar kullanılabilirlik ve Esneklik Niteliklerinin değişimi

Farklı Versiyonlar için

- Etkinlik,
- Genişletilebilirlik,
- İşlevsellik,
- Tekrar kullanılabilirlik
- Esneklik

Niteliklerinin 9 farklı versiyon için bu metriklerin versiyona bağlı olarak değişimi yukarıda gösterilmiştir. Bu hesaplamalarda yer alan niteliklere özgü sayısal veriler y ekseninde yer alırken versiyonun numaraları ise en altta x ekseni düzleminde yer almaktadır.Sezgisel olarak baktığımızda ise bu sayısal verilerin en başta artarak yükselmesi beklenmektedir fakat esneklik en başta ufak miktar artmış olsa bile sonradan düşüşe geçmiş ve geri kalan tüm veriler aşağı ve yukarı şeklinde ilerlemelerine rağmen genel yönelim aşağı yöne doğru olmuştur. Bu beklenmedik düşüşün sebebi uygulamanın ilk versiyonun yeterince nesneyeli yönelik olmaması , clean code şeklinde yazılmaması ve tekrarlanan fonksiyon ve benzeri öğelerin düzenlenmemesi ilk başta karışık bir versiyon ile çıktıktan sonra diğer versiyonlarda bu durumun iyileştirilmesi şeklinde yorumlanabilir.

Aşağıda farklı Versiyonlar için

- Anlaşılabilirlik

Niteliğinin 9 farklı versiyon için bu metriklerin versiyona bağlı olarak değişimi yukarıda gösterilmiştir. Bu hesaplamalarda yer alan niteliklere özgü sayısal veriler y ekseninde yer alırken versiyonun numaraları ise en altta x ekseni düzleminde yer almaktadır. Sezgisel olarak bakıldığında anlaşılabilirlik metriğinin ilk başta azalması sonra artması beklenmektedir fakat ilk grafiktede yaşanan bu beklenmedik durum burada da yaşanarak tahminlerimizin doğuluğunu göstermektedir. Anlaşılabilirlik ilk başta projenin karmşıklığı sebebi ile düşük olsada sonradan bu değerler yukarı yönlü bir ivme göstermiştir. V1.3.0-V1.3.2-V1.3.4 te ufak düşüşler yaşasada genel olarak yükselmiştir.

Şekil 4: Farklı Versiyonlar için Anlaşılabilirlik Niteliğinin Değişimi

Versiyon	Encapsulation	Polymorfism	Hieracies	Messaging	Design Size	Composotion	Abstract	Inheritance	Cohesion	Complexity	Coupling
V.1.0.0	1	1	1	1	1	1	1	1	1	1	1
V.1.10	1,75	0,73	1,75	1	0,23	1	1,75	0,18	0,75	1,02	1,08
V.1.2.0	1	0,73	1	1	0,23	1,03	0,55	0,4	0,75	0,99	1
V.1.3.0	1	1	1	0,96	0,27	0,95	0,12	0,18	0.5	0,99	1
V.1.3.1	1	0,77	1	1,07	0,25	1,02	0,09	0,55	1,5	0,98	1
V.1.3.2	1	0,83	1	0,95	0,24	1,06	0,11	0,55	1.25	0,95	1
V.1.3.3	1	0,88	1	1,05	0,25	1	0,14	0,18	1,5	0,99	1
V.1.3.4	1	0,82	1	0,96	0,21	1,03	0,21	0,18	1,25	1	1
V.1.4.0	1	0,83	1	0,98	0,21	1,02	0,2	0,55	1	0,99	1

Versiyon	Tekrar Kullanılabilirlik	Esneklik	İşlevsellik	Genişletilebilirlik	Etkinlik	Anlaşılabilirlik
V.1.0.0	1	1	1	1	1	-1
V.1.10	0,5325	1,0325	0,9956	1	1	-0,99
V.1.2.0	0,5525	0,88	0,7412	0,79	1,082	-0,7623
V.1.3.0	0,49	0,975	0,75	0,34	0,742	-0,5775
V.1.3.1	0,785	0,895	0,7662	0,15	0,65	-0,6204
V.1.3.2	0,6575	0,945	0,8576	0,205	0,686	-0,1947
V.1.3.3	0,775	0,94	0,8166	0,245	0,71	-0,2904
V.1.3.4	0,6475	0,925	0,8708	0,1	0,64	-0,2508
V.1.4.0	0,605	0,925	0,8122	0,105	0,648	-0,3267

VI. SONUÇ

Bu çalışmada windows platform için c# programala dili kullanılarak hazırlanan bir mobil uygulama ele alınmıştır. Bu uygulamanın 1 yıl içerisinde geliştirilip açık kaynak kodlu olarak yayınlanan 9 farklı versiyonu QMOOOD modeli kullanılarak incelenmiştir. Sonuç olarak değerlendirdiğimizde ise ilgili modelin kullanılan yaklaşımın sezgisel olarak beklenen değerlerden farklı bir çıktı verdiği fakat kendi içinde tutarlı olduğu gözlemlenmiştir. Tasarımın bir uygulama içerisinde ki değişimi nesneye yönelik dayalı tasarım kriterleri göz önüne alınarak hazırlanan uygulamanın doğruluğunu göstermiştir. Ayrıca çıkan kalite niteliklerinin değişim karakteristiğine bakıldığı zaman projenin geliştirilmesi sırasında yapılan hata düzeltme ve projeyi düzeltme esnasında yapılan sürümler güncellemelerinde nesneye dayalı programlamaa presinsiplerinden uzaklaşdığı yada dikkate alınmadığı söylenebilir. Esas geliştirme sonrasında projenin büyük çoğunlunun düzeltme ve onarma işlemi ile geçtiğini de vurgulanabilir. Proje ile ilgili geliştirmelerin bir anda yapılıp sonrasında ise düzeltmeye ayrılması nesneye dayalı proje geliştirme presiplerine uyulmadığı konusunda bir eleştiri getirilebilir.

KAYNAKÇA

[1] S. Chidamber and C. Kemerer, “A metrics suite for object oriented-design,” IEEE Transactions 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 SoftwareEngineering, vol. 28, no. 1, pp.4-17, (2002).

[3] Pulumi Modern Infrastructure as Code, https://www.pulumi.com

[4] Improve your .NET code quality with NDepend, https://www.ndepend.com

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

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

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

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

[9] M. Lanza and R. Marinescu, Object-Oriented Metrics in Practice: Us-ingSoftware Metrics to Characterize, Evaluate, and Improve the DesignofObject-Oriented Systems. Springer, Heidelberg (2006).