Nesneye Dayalı Kalite Modelleri

ISO/IEC'nin yazılım kalitesi modelleri belli bir yazılım tasarım yöntemine bağlı olmadan yazılımların kalitelerini ele alırlar.

Ayrıca bu modellerde, alt düzeydeki metriklerin üst düzeydeki kalite niteliklerine sayısal olarak (normalizasyon, ağırlıklar) nasıl bağlanacağı yer almaz.

Bu tür genel yazılım kalitesi modellerinin yanı sıra doğrudan nesneye dayalı yazılımların kalite özelliklerini değerlendirmeyi hedefleyen modeller ve metrikler de bulunmaktadır.

Bu bölümde Bansiya ve Davis * tarafından oluşturulan bir model olan QMOOD (Quality Model for Object-Oriented Design) ele alınacaktır.

Katmanlı yapıdaki modelde alt düzeydeki metrikler ile üst düzeydeki kalite nitelikleri arasındaki ilişkiler kurulmuştur.

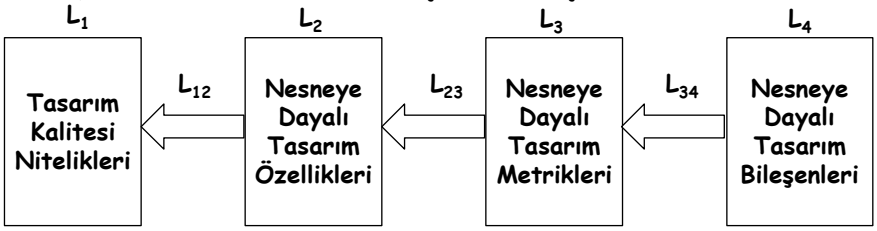
---

* J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 4–17, Jan. 2002.

QMOOD: Quality Model for Object-Oriented Design

Hiyerarşik katmanlı kalite modeli benimsenmiştir.

Model, dört katman ve aralarındaki ilişkilerden oluşmaktadır.



L_1: Tasarım Kalitesi Nitelikleri (*Design Quality Attributes*)
Asıl ölçülmek istenen üst düzey niteliklerdir.

L_2: Nesneye Dayalı Tasarım Özellikleri (*Object-Oriented Design Properties*)
Üst düzey kalite niteliklerini etkileyen nesneye dayalı tasarım özellikleri

L_3: Nesneye Dayalı Tasarım Metrikleri (*Object-Oriented Design Metrics*)
Nesneye dayalı tasarım özelliklerine değer atamak (ölçmek) için kullanılacak olan nesneye dayalı metrikler

L_4: Nesneye Dayalı Tasarım Bileşenleri (*Object-Oriented Design Components*)

Düzey 1 (L_1): Tasarım Kalitesi Nitelikleri (*Design Quality Attributes*)

ISO/IEC 9126'daki işlevsellik (*functionality*), güvenilirlik (*reliability*), verimlilik (*efficiency*), kullanışlılık (*usability*), bakım kolaylığı (*maintainability*) ve taşınabilirlik (*portability*) temel alınmıştır.

Tasarımdan çok gerçekleme (*implementation*) ile ilgili olan güvenilirlik ve kullanışlılık listeden çıkartılmıştır.

Taşınabilirlik (*portability*), genişletilebilirlik (*extendibility*) olarak değiştirilmiştir.

Verimlilik (*efficiency*), etkinlik (*effectiveness*) olarak değiştirilmiştir.

Bakım kolaylığı, anlaşılırlık (*understandability*) olarak değiştirilmiştir.

Tekrar kullanılabilirlik (*reusability*) eklenmiştir.

Esneklik (*flexibility*) eklenmiştir.

QMOOD Tasarım Kalitesi Nitelikleri (L1):

- İşlevsellik (*functionality*)
- Etkinlik (*effectiveness*)
- Anlaşılırlık (*understandability*)
- Genişletilebilirlik (*extendibility*)
- Tekrar kullanılabilirlik (*reusability*)
- Esneklik (*flexibility*)

QMOOD tasarım kalitesi niteliklerinin anlamları:

Quality Attribute	Definition
Reusability	Reflects the presence of object-oriented design characteristics that allow a design to be reapplied to a new problem without significant effort.
Flexibility	Characteristics that allow the incorporation of changes in a design. The ability of a design to be adapted to provide functional related capabilities.
Understandability	The properties of the design that enable it to be easily learned and comprehended. This directly relates to the complexity of the design structure.
Functionality	The responsibilities assigned to the classes of a design, which are made available by the classes through their public interfaces.
Extendibility	Refers to the presence and usage of properties in an exiting design that allow for the incorporation of new requirements in the design.
Effectiveness	This refers to a design's ability to achieve the desired functionality and behavior using object-oriented design concepts and techniques.

Düzey 2 (L_2): Nesneye Dayalı Tasarım Özellikleri
(*Object Oriented Design Properties*)

Tasarım özellikleri; bir yazılımdaki tasarım birimlerinin (sınıflar, metotlar vb.) iç niteliklerini, işlevlerini ve aralarındaki ilişkileri ele alarak doğrudan değerlendirilebilecek somut kavramlardır.

QMOOD'taki tasarım özellikleri:

Bu özellikler (11 adet) bir veya daha fazla tasarım metriği ile değerlendirilebilecek şekilde seçilmiştir.

Design Property	Definition
Design Size	A measure of the number of classes used in a design.
Hierarchies	Hierarchies are used to represent different generalization-specialization concepts in a design. It is a count of the number of non-inherited classes that have children in a design.
Abstraction	A measure of the generalization-specialization aspect of the design. Classes in a design which have one or more descendants exhibit this property of abstraction.
Encapsulation	Defined as the enclosing of data and behavior within a single construct. In object-oriented designs the property specifically refers to designing classes that prevent access to attribute declarations by defining them to be private, thus protecting the internal representation of the objects.

QMOOD'taki tasarım özellikleri (devamı):

Design Property	Definition
Coupling	Defines the interdependency of an object on other objects in a design. It is a measure of the number of other objects that would have to be accessed by an object in order for that object to function correctly.
Cohesion	Assesses the relatedness of methods and attributes in a class. Strong overlap in the method parameters and attributes types are an indication of strong cohesion.
Composition	Measures the "part-of", "has", "consists-of" or "part-whole" relationships, which are aggregation relationships in an object-oriented design.
Inheritance	A measure of the "is-a" relationship between classes. This relationship is related to the level of nesting of classes in an inheritance hierarchy.
Polymorphism	The ability to substitute objects whose interfaces match for one another at run-time. It is a measure of services that are dynamically determined at runtime in an object.
Messaging	A count of the number of public methods that are available as services to other classes. This is a measure of the services that a class provides.
Complexity	A measure of the degree of difficulty in understanding and comprehending the internal and external structure of classes and their relationships.

Düzey 3 (L3): Nesneye Dayalı Tasarım Metrikleri
(*Object Oriented Design Metrics*)

- Tasarım metrikleri yazılımın tasarım aşamasında elde edilebilirler.
- Bu metrikler tasarım özelliklerinin değerlendirilmesinde doğrudan kullanılırlar.

QMOOD'taki tasarım metrikleri:

Bir kısmı önceki çalışmalardan alınmış, bir kısmı yeni oluşturulmuştur.

Metric	Name	Description
DSC	Design Size in classes	This metric is a count of the total number of classes in the design.
NOH	Number of Hierarchies	This metric is a count of the number of class hierarchies in the design.
ANA	Average Number of Ancestors	This metric value signifies the average number of classes from which a class inherits information. It is computed by determining the number of classes along all paths from the "root" class(es) to all classes in an inheritance structure.
DAM	Data Access Metric	This metric is the ratio of the number of private (protected) attributes to the total number of attributes declared in the class. A high value of DAM is desired (Range 0 to 1).

QMOOD'taki tasarım metrikleri (devamı):

Metric	Name	Description
DCC	Direct Class Coupling	This metric is a count of different number of classes that a class is directly related to. The metric includes classes that are directly related by attribute declarations and message passing (parameter) in methods.
CAM	Cohesion Among Methods of Class	This metric computes the relatedness among methods of a class based upon the parameter list of methods. The metric is computed using the summation of the intersection of parameters of a method with the maximum independent set of all parameter types in the class. A metric value close to 1.0 is preferred (Range 0 to 1).
MOA	Measure of aggregation	This metric measures the extent of part-whole relationship, realized by using attributes. The metric is a count of the number of data declarations whose types are user defined classes.

QMOOD'taki tasarım metrikleri (devamı):

Metric	Name	Description
MFA	Measure of Functional Abstraction	This metric the ratio of the number of methods inherited by a class to the total number of methods accessible by member methods of the class (Range 0 to 1).
NOP	Number of Polymorphic methods	This metric is a count of the methods that can exhibit polymorphic behavior. In C++: virtual methods.
CIS	Class interface Size	This metric is a count of the number of public methods in a class.
NOM	Number of methods	This metric is a count of all the methods defined in a class.

Düzey 4 (L4): Nesneye Dayalı Tasarım Bileşenleri (*Object Oriented Design Components*)

Nesneye dayalı bir tasarımın yapısını belirleyen ve tanımlanabilen en temel bileşenler; nesneler, sınıflar ve aralarındaki ilişkilerdir.

Buna göre nesneye dayalı bir tasarımın değerlendirilmesinde aşağıdaki bileşenler üzerinde çalışılabilir (metrikler elde edilebilir).

Sınıflar (nesneler), nitelikler, metotlar, ilişkiler, sınıf hiyerarşileri.

Bileşenlerin kaliteyi etkileyen özellikleri:

- Nitelikler:

İsim, public/private/protected, static/constant, başlangıç değeri atanmış mı, boyut, tip, değer/işaretçi

- Metotlar:

İsim, public/private/protected, parametre sayısı ve tipleri, static/dynamic, virtual/non-virtual, constructor/destructor, constant, inline

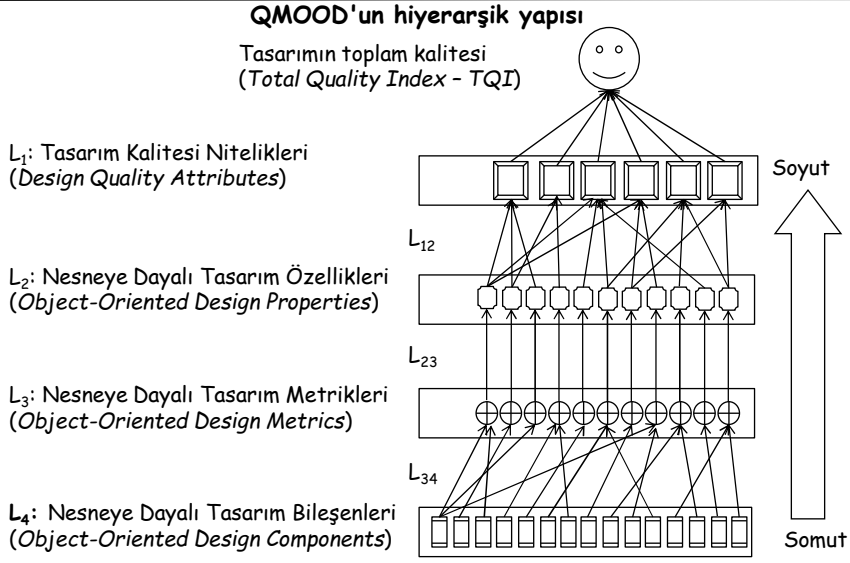
- Sınıflar:

İsim, erişim tipi (Java), türetim tipi, taban sınıf mı, şablon mu, türetilmiş sınıf mı, türetim ağacındaki sırası, alt sınıf sayısı, nitelikleri, metotları, diğer sınıflarla ilişkisi, metotları arasındaki uyum

Tasarım Metrikleri ile Tasarım özellikleri arasındaki ilişki (L23):

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

QMOOD'un hiyerarşik yapısı



ISO/IEC 25020 Software quality measurement reference model (SPQM_RM) benzerdir (bkz. 2-37-2.39).

Tasarım özellikleri ile kalite nitelikleri arasındaki ilişki (L_{12}):

Birçok doküman incelenerek ve deneyimler kullanılarak sezgisel olarak belirlenmiştir.

	Reusability	Flexibility	Understandability	Functionality	Extendibility	Effectiveness
Design size	↑		↓	↑		
Hierarchies				↑		
Abstraction			↓		↑	↑
Encapsulation		↑	↑			↑
Coupling	↓	↓	↓		↓	
Cohesion	↑		↑	↑		
Composition		↑				↑
Inheritance					↑	↑
Polymorphism		↑	↓	↑	↑	↑
Messaging	↑			↑		
Complexity			↓			

Tasarım özellikleri ile kalite nitelikleri arasındaki ağırlıklı ilişki:

Ağırlıklar toplamları +1.0 ya da -1.0 olacak şekilde seçilmiştir.

Reusability:
+0.5*Design size -0.25*Coupling +0.25*Cohesion +0.5*Messaging

Flexibility:
+0.25*Encapsulation -0.25*Coupling +0.5*Composition +0.5Polymorphism

Undarstandibility:
-0.33*Design size -0.33*Abstraction +0.33*Encapsulation
-0.33*Coupling +0.33*Cohesion -0.33*Polymorphism -0.33*Complexity

Functionality:
+0.22*Design size +022*Hierarchies +0.12*Cohesion +0.22*Polymorphism
+0.22*Messaging

Extendibility:
+0.5*Abstraction -0.5*Coupling +0.5*Inheritance +0.5*Polymorphism

Effectiveness:
+0.2*Abstraction +0.2*Encapsulation +0.2*Composition +0.2*Inheritance
+0.2*Polymorphism

Tasarım özellikleri ile kalite nitelikleri arasındaki ağırlıklı ilişki:

	Reusability	Flexibility	Understandability	Functionality	Extendibility	Effectiveness
Design size	+0.5		-0.33	+0.22		
Hierarchies				+0.22		
Abstraction			-0.33		+0.5	+0.2
Encapsulation		+0.25	+0.33			+0.2
Coupling	-0.25	-0.25	-0.33		-0.5	
Cohesion	+0.25		+0.33	+0.12		
Composition		+0.5				+0.2
Inheritance					+0.5	+0.2
Polymorphism		+0.5	-0.33	+0.22	+0.5	+0.2
Messaging	+0.5			+0.22		
Complexity			-0.33			
Toplam	+1.0	+1.0	-1.0	+1.0	+1.0	+1.0

Tasarımın toplam kalitesi (*Total Quality Index – TQI*) :
Altı kalite niteliğinin değerleri toplanarak o projenin toplam kalitesini gösteren toplam kalite indeksi (*Total Quality Index – TQI*) hesaplanır.

QMOOD Modelin geçerliliğinin sınanması (*validation*)

Modelin geçerliliği iki aşamada sınanmıştır.

1. Kalite niteliklerinin geçerliliği
2. Toplam kalite değerinin geçerliliği

1. Modeldeki Kalite niteliklerinin geçerliliğinin sınanması:

Sınama için iki yazılım grubu kullanılmıştır.

a. Microsoft Foundation Classes (MFC)
b. Borland Object Windows Library (OWL)

Her iki yazılım platformu da Windows tabanlı görsel yazılımlar geliştirmekte kullanılmıştır.

MFC'nin 5 sürümü, OWL'nin 4 sürümü incelenmiştir.

Windows Application Frameworks	Release Date
Microsoft Foundation Classes, MFC 1.0	1992, with C/C++ 7.0 and Windows NT
Microsoft Foundation Classes, MFC 2.0	Early 1993, with Visual C++ version 1.0 for Windows
Microsoft Foundation Classes, MFC 3.0	Late 1994, with Visual C++ version 2.0
Microsoft Foundation Classes, MFC 4.0	Late 1995, with Visual C++ 4.0
Microsoft Foundation Classes, MFC 5.0	Early 1997, with Visual C++ 5.0
Object Windows Library, OWL 4.0	Mid 1994, with Borland C++ 4.0
Object Windows Library, OWL 4.5	Late 1995, with Borland C++ 4.5
Object Windows Library, OWL 5.0	Mid 1996, with Borland C++ 5.0
Object Windows Library, OWL 5.2	Mid 1997, with Borland C++ 5.2

Yazımların kalitesi ile ilgili beklentiler (varsayımlar)

Aşağıdaki yorumlar gözlemlere dayanmaktadır:

- Bir yazılımın yeni sürümünün yayımlanmasının iki temel nedeni vardır:

1. Yazılıma yeni işlevler, özellikler, yetenekler katmak
2. Belirlenen hataları düzeltmek

- Yeni bir yazılımın erken sürümleri genellikle yazılıma yeni yetenekler kazandırmak için yayımlanır.
- Erken sürümlerde kullanışlılık ve kullanıcı dostu özellikleri de artırılır.
- Bu nedenlerle erken sürümlerde yapısal değişiklikler büyük olur ve bir önceki sürüme göre toplam kalitedeki iyileşme de büyük olur.
- Yazılım olgunlaştıktan sonraki sürümlerde amaç genellikle hataları düzeltmek, yazılımın güvenirliliğini ve sağlamlığını artırmaktır.
- Geç sürümlerin bir amacı da yazılımın karmaşıklığını azaltmak olmaktadır.
- Bu nedenlerle olgun bir yazılımın sürümleri arasındaki kalite artışı az olmaktadır.

Modelin üretmesi beklenen sonuçlar

Yazılımların sürümleri ile ilgili gözlemlere göre modelin geçerli olabilmesi için aşağıdakine yakın sonuçlar üretmelidir.

- Tekrar kullanılabilirlik (*reusability*), esneklik (*flexibility*), işlevsellik (*functionality*), genişletilebilirlik (*extendibility*), etkinlik (*effectiveness*) her sürümde belli bir oranda artmalı.
- Özellikle işlevsellik (*functionality*) ve etkinlik (*effectiveness*) ilk sürümlerde daha fazla artmalı.
- İlk sürümlerde anlaşılırlık (*understandability*) düşmeli. Çünkü bu sürümlerde yazılıma yeni özelikler eklemek için sisteme bir çok sınıf, metot eklenecek.
- Yazılım olgunlaştıkça anlaşılırlık artmaya başlamalı.

Elde edilen metrik değerleri:

Bazı metrikler tüm yazılımla ilgilidir; örneğin design size.

Bazı metrikler sadece tek bir sınıfla ilgilidir; örneğin coupling, polymorphism.

Bu metrikler, ortalama hesabı ile **program düzeyine dönüştürülmüştür**.

METRIC	MFC 1.0	MFC 2.0	MFC 3.0	MFC 4.0	MFC 5.0	OWL 4.0	OWL 4.5	OWL 5.0	OWL 5.2
Design Size	72.00	92.00	132.00	206.00	233.00	82.00	142.00	357.00	356.00
Hierarchies	1.00	1.00	1.00	5.00	6.00	12.00	16.00	29.00	27.00
Abstraction	1.68	2.11	2.45	2.33	2.30	1.00	0.96	1.60	1.55
Encapsulation	0.61	0.48	0.54	0.56	0.58	0.71	0.66	0.62	0.63
Modularity	0.80	0.90	0.92	0.97	1.02	1.30	1.30	1.40	1.40
Coupling	6.03	7.59	9.02	9.33	8.94	1.90	2.30	4.70	4.71
Cohesion	0.20	0.15	0.16	0.15	0.16	0.39	0.41	0.38	1.00
Aggregation	0.26	0.91	1.39	1.38	1.45	0.37	0.70	1.40	1.36
Inheritance	0.07	0.35	0.43	0.50	0.49	0.43	0.39	0.45	0.45
Polymorphism	6.83	18.20	19.30	28.30	28.60	1.90	3.10	7.00	6.78
Messaging	44.60	75.60	95.00	114.00	109.00	28.00	24.00	37.00	54.60
Complexity	56.50	102.00	126.00	150.00	144.00	30.00	29.00	56.00	36.80

İlk sürümlere göre normalize edilmiş metrik değerleri:

Metriklerin değer aralıkları farklı olduğundan bunları aynı denklem içinde kullanmak mümkün değildir.

Her yazılım kendi içinde ayrı ayrı normalize edilmiştir.

MFC 1.0'a göre ve OWL 4.0'a göre

METRIC	MFC 1.0	MFC 2.0	MFC 3.0	MFC 4.0	MFC 5.0	OWL 4.0	OWL 4.5	OWL 5.0	OWL 5.2
Design Size	1.00	1.28	1.83	2.86	3.24	1.00	1.73	4.35	4.34
Hierarchies	1.00	1.00	1.00	5.00	6.00	1.00	1.33	2.42	2.25
Abstraction	1.00	1.26	1.46	1.39	1.37	1.00	0.97	1.56	1.55
Encapsulation	1.00	0.79	0.88	0.92	0.94	1.00	0.94	0.88	0.89
Modularity	1.00	1.13	1.15	1.21	1.28	1.00	1.01	1.04	1.04
Coupling	1.00	1.26	1.50	1.55	1.48	1.00	1.21	2.53	2.52
Cohesion	1.00	0.74	0.81	0.79	0.83	1.00	1.00	1.00	1.00
Aggregation	1.00	3.46	5.27	5.23	5.49	1.00	1.90	3.85	3.72
Inheritance	1.00	5.39	6.55	7.61	7.57	1.00	0.92	1.06	1.06
Polymorphism	1.00	2.66	2.83	4.14	4.19	1.00	1.66	3.71	3.59
Messaging	1.00	1.81	2.23	2.65	2.55	1.00	0.95	1.83	1.80
Complexity	1.00	1.70	2.13	2.56	2.44	1.00	0.87	1.34	1.33

Hesaplanan Kalite Nitelikleri (normalize metrik değerleri kullanılmıştır)

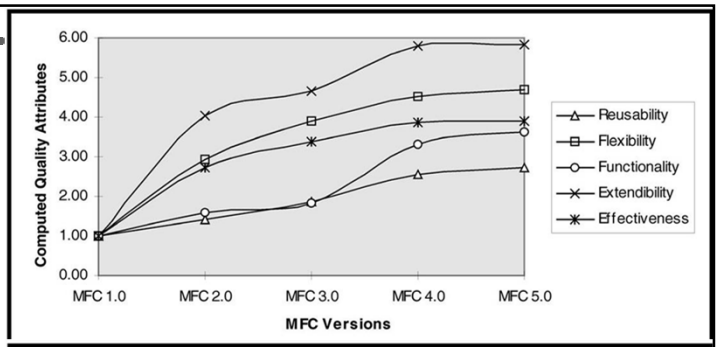
Quality Index	MFC 1.0	MFC 2.0	MFC 3.0	MFC 4.0	MFC 5.0	*OWL 4.0*	*OWL 4.5*	*OWL 5.0*	*OWL 5.2*
Reusability	1.00	1.41	1.86	2.57	2.73	*1.00*	*1.29*	*2.71*	*2.69*
Flexibility	1.00	2.94	3.89	4.53	4.71	*1.00*	*1.72*	*3.37*	*3.24*
Understandability	-0.99	-2.18	-2.66	-3.56	-3.61	*-0.99*	*-1.48*	*-3.83*	*-3.77*
Functionality	1.00	1.57	1.83	3.32	3.61	*1.00*	*1.37*	*2.83*	*2.76*
Extendibility	1.00	4.03	4.67	5.80	5.82	*1.00*	*1.17*	*1.90*	*1.84*
Effectiveness	1.00	2.71	3.40	3.86	3.91	*1.00*	*1.28*	*2.21*	*2.16*

Sonuçların yorumlanması:

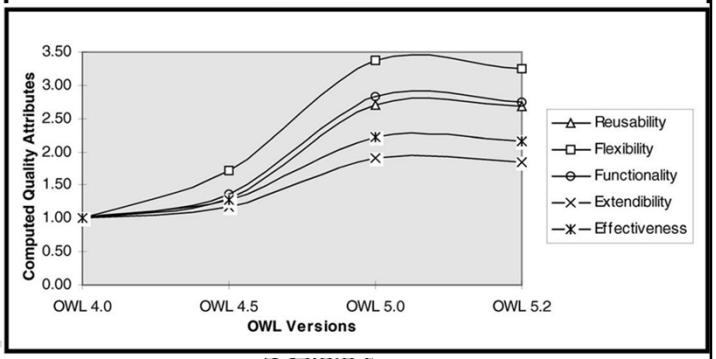
- Beklendiği gibi tekrar kullanılabilirlik (*reusability*), esneklik (*flexibility*), işlevsellik (*functionality*), genişletilebilirlik (*extendibility*), etkinlik (*effectiveness*) her sürümde belli bir oranda artıyor.
- İlk sürümlerdeki artış oranı daha fazla (Bkz. 5.21 şekiller)
- Anlaşılırlık beklenenin dışında MFC'nin tüm sürümlerinde azalıyor.

  5.0 sürümündeki azalma çok az olduğu için yazılımın bu sürümde olgunlaşmaya başladığı düşünülebilir.
- OWL 5.2 sürümünde anlaşılırlık bir önceki sürüme göre iyileşmeye başlıyor.

MFC için elde edilen kalite nitelikleri değerleri



OWL için elde edilen kalite nitelikleri değerleri



2. Modelin toplam kalite ile ilgili geçerliliğinin sınanması

Sınama Ortamı:

- Farklı konularda, farklı amaçlarla yapılan tasarımların karakteristikleri de farklı olacaktır.

  Bu nedene aralarında karşılaştırma yapılacak olan tasarımların aynı amaçlarla gerçeklenmiş olması gerekir.

- Sınamaları yapmak için orta boyutlarda (10-29 sınıf) bir C++ projesi seçilmiştir.

  Bu boyutlardaki bir projeyi insan gözüyle de değerlendirmek mümkündür.

- Proje olarak 14 farklı kişiye "COOL" adlı sanal bir programlama dili için yorumlayıcı (*interpreter*) yazdırılmıştır.

  Her yazılım iki sürümden oluşmaktadır. Önce belli gereksinimler verilere 1.0 sürümü yazdırılmış ardından ek istekler verilerek 2.0 sürümü yazdırılmıştır.

  Böylece ortaya kalitesi ölçülecek 14 farklı proje çıkmıştır.

- Modelin geçerliliğini sınamak için 13 kişiden oluşan bir değerlendirici grubu kullanılmıştır.

  Değerlendiricilerin ticari yazılımlar geliştirme ve nesneye dayalı tasarım konusunda iki ile yedi yıl arasında deneyimleri bulunmaktadır.

Sınama Yöntemi:

- Her değerlendiricinin tasarımlardaki 6 kalite niteliğini (*Reusability, Flexibility, Understandability, Functionality, Extendibility, Effectiveness*) ölçen kendilerine özgü sezgisel bir yöntem (*heuristic*) oluşturması isteniyor.
- Değerlendiriciler kendilerine verilen 14 projeyi kendi yöntemleri ile değerlendirmişlerdir.

  Bu değerlendirmede her proje için 6 kalite niteliğini kendi yöntemleri ile ölçüp bu 6 değeri toplayarak o proje için toplam kalite puanını (*Total Quality Score – TQS*) hesaplamışlardır.
- Aynı kalite nitelikleri QMOOD yöntemiyle de hesaplanmıştır.

  Önce metrikler elde edilmiş, ardından metrik değerleri modeldeki katsayılarla çarpılarak kalite niteliklerinin değerleri hesaplanmıştır.

  Son olarak da kalite niteliklerinin değeri toplanarak o projenin toplam kalitesini gösteren toplam kalite indeksi (*Total Quality Index – TQI*) hesaplanmıştır.
- Projeler elde ettikleri puanlara göre kendi aralarında sıralanmıştır.

  Yüksek kaliteli proje 1. sırada, düşük kaliteli proje 14. sırada yer almıştır.

Sıralama sonuçları:

PROJECT NUMBER	QMOOD RANKING	EVALUATOR RANKINGS												
		1	2	3	4	5	6	7	8	9	10	11	12	13
1	*13*	12	9	11	11	13	12	12	13	13	13	11	8	9
2	*9*	13	12	12	12	12	6	13	6	10	12	12	12	10
3	*3*	2	6	2	1	7	1	4	1	2	7	4	2	1
4	*12*	9	7	9	10	2	9	10	9	6	11	10	5	6
5	*1*	1	1	1	2	1	7	1	2	1	2	2	3	2
6	*5*	8	11	10	7	9	11	6	12	11	9	3	11	12
7	*11*	4	13	13	13	8	13	11	8	12	4	1	13	8
8	*8*	3	8	9	8	11	10	9	10	8	3	7	6	11
9	*7*	5	10	7	9	10	3	5	4	5	6	6	7	7
10	*2*	6	2	3	3	4	5	2	3	3	1	5	1	5
11	*4*	11	3	5	6	5	8	3	11	9	8	8	10	13
12	*6*	7	5	6	4	3	2	8	5	7	5	9	9	3
13	*10*	10	4	4	5	6	4	7	7	4	10	13	4	4
14	*14*	14	14	14	14	14	14	14	14	14	14	14	14	14

- 14 numaralı proje konusunda tüm değerlendiriciler aynı fikirdedir.
- 1, 2, 3, 5, 7 ve 10 numaralı projeler konusunda büyük ölçüde uzlaşma vardır.
- 13 numaralı proje konusunda anlaşmazlık vardır.
  Yazarlar bunu projedeki sınıf sayısının az olmasına bağlıyorlar ancak 1, 2, 14 numaralı projelerde daha az sınıf bulunmaktadır.

QMOOD ile elde edilen değerlerin ayrıntıları:

Metrik değerleri:

METRIC	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Design Size	12.0	10.0	15.0	22.0	24.0	26.0	29.0	23.0	20.0	20.0	24.0	25.0	12.0	3.00
Hierarchies	1.0	1.0	1.0	1.0	1.0	2.0	2.0	4.0	1.0	1.0	2.0	1.0	1.0	0.00
Abstraction	0.7	1.2	2.1	1.2	1.4	0.7	0.9	1.0	1.3	1.4	0.7	1.3	1.3	0.00
Encapsulation	1.0	0.6	0.8	0.5	0.9	0.8	1.0	0.8	0.8	1.0	1.0	0.6	0.7	0.63
Modularity	0.7	0.5	0.4	0.4	0.5	0.4	0.6	0.5	0.4	0.4	0.3	0.3	0.4	0.26
Coupling	1.5	0.8	0.0	1.0	0.6	1.2	1.2	1.8	1.8	0.9	1.1	1.1	1.7	0.67
Cohesion	0.5	0.6	0.6	0.4	0.7	0.6	0.4	0.5	0.3	0.5	0.6	0.6	0.5	0.53
Aggregation	0.3	0.0	0.0	0.4	0.1	0.8	0.2	1.1	0.3	0.1	0.6	0.6	0.1	0.67
Inheritance	0.2	0.4	0.8	0.5	0.7	0.6	0.4	0.5	0.5	0.7	0.5	0.2	0.4	0.00
Polymorphism	2.4	3.8	1.5	1.5	2.5	2.3	2.8	3.3	5.1	4.3	2.3	3.7	5.6	0.00
Messaging	8.9	6.9	11.0	9.6	7.5	9.0	8.4	14.0	17.0	12.0	8.1	7.2	9.8	5.30
Complexity	8.9	7.7	9.9	9.6	7.5	9.0	8.4	14.0	17.0	12.0	8.1	7.2	9.8	5.30

1-14 arası normalize edilerek sıralanan metrik değerleri:

Metrics	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Design Size	4	2	5	8	11	13	14	9	7	7	11	12	4	1
Hierarchies	10	10	10	10	10	13	13	14	10	10	13	10	10	1
Abstraction	3	8	14	8	13	2	5	6	11	13	4	11	11	1
Encapsulation	14	3	8	1	10	7	14	6	9	14	14	2	5	4
Modularity	14	11	6	5	10	4	13	12	7	9	2	3	9	1
Coupling	11	4	1	6	2	10	10	14	14	5	8	8	12	3
Cohesion	5	11	13	3	14	10	2	5	1	6	12	10	8	8
Aggregation	7	2	2	9	5	13	6	14	8	3	11	10	4	12
Inheritance	3	5	14	9	13	11	4	7	9	12	10	2	6	1
Polymorphism	6	11	3	3	7	5	8	9	13	12	5	10	14	1
Messaging	7	2	11	9	4	8	6	13	14	12	5	3	10	1
Complexity	7	4	11	9	3	8	6	13	14	12	5	2	10	1

Bu normalizasyon ile metrik değerleri arasındaki orantısal değişim kaybolmuştur.

Değerler, projeler arasında sadece sıralama yapmaktadır.

QMOOD ile elde edilen değerlerin ayrıntıları (devamı):

QMOOD ile ölçülen toplam kalite indeksi (Total Quality Index – TQI):

Quality	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Reusability	4.0	3.7	11.0	7.7	10.5	10.5	8.0	8.7	7.2	9.7	9.0	8.0	6.0	2.2
Flexibility	7.2	6.2	4.2	4.7	8.0	8.2	8.0	9.5	9.2	9.7	9.5	8.5	7.2	6.7
Understandability	-3.9	-4.9	-4.2	-9.9	-3.9	-6.9	-8.9	-13.2	-16.1	-9.5	-2.3	-10.2	-12.5	-1.6
Functionality	6.5	6.8	7.9	6.9	8.7	9.7	9.2	10.5	9.8	9.7	8.9	8.9	9.3	1.8
Extendibility	0.5	10.0	15.0	7.0	15.5	4.0	3.5	4.0	9.5	16.0	5.5	7.5	9.5	0.0
Effectiveness	6.6	5.8	8.2	6.0	9.6	7.6	7.4	8.4	10.0	10.8	8.8	7.0	8.0	3.8
TQI	**20.9**	**27.6**	**42.1**	**22.5**	**48.3**	**33.2**	**27.2**	**27.9**	**29.6**	**46.4**	**39.4**	**29.6**	**27.5**	**16.2**

TQI değerlerine göre projeler sıralanarak 5.24'teki tablo oluşturulmuştur.

Burada TQI değerlerinin sırasal ölçekte (*ordinal scale*) ölçüm yapmaktadır.

Metrik değerlerini aynı aralığa getirmek için yapılan 1-14 arası normalizasyon işlemi metrikleri sırasal ölçeğe getirmiştir.

Spearman sıralama korelasyonu katsayı sınaması (*Spearman's rank correlation coefficient test*):

QMOOD ile yapılan sıralama ile değerlendiricilerin yaptığı sıralama arasındaki korelasyonun anlamlılığını sınamak için kullanılmıştır.

Spearman sıralama korelasyonu katsayısının hesaplanması:

E_1 ve E_2, n adet elaman ile ilgili yapılan bağımsız, sıralama değerlendirmeleridir.

Bu değerlendirmeler (E_1 ve E_2) elemanlara 1'den n'ye kadar sayılar atarlar.

d farkı E_1 ve E_2 değerlendirmelerinin aynı elemana atadıkları sıralama değerinin farkıdır.

Spearman katsayısı:

$$r_s = 1 - \frac{6 \sum d^2}{n(n^2 - 1)} \qquad -1.00 \leq r_s \leq +1.00$$

Sıralama korelasyonu sonuçları:

QMOOD'un yaptığı sıralama 13 değerlendiricin yaptığı sıralamalarla ayrı ayrı karşılaştırılmıştır.

r_s katsayısı 0.55'ten büyükse anlamlı bir korelasyon olduğu kabul edilmiştir.

Değerlendiriciler:

	1	2	3	4	5	6	7	8	9	10	11	12	13
Σd^2	180	146	91	68	198	196	42	154	142	136	168	214	260
r_s	0.60	0.68	0.80	0.85	0.56	0.57	0.91	0.66	0.69	0.70	0.63	0.53	0.43
$r_s > 0.55$	√	√	√	√	√	√	√	√	√	√	√	X	X

12 ve 13 numaralı değerlendiricilerin yaptığı sıralama ile QMOOD'un sıralaması arasındaki korelasyon kabul edilebilir eşik değerinin altında kalmıştır.

Bu iki değerlendiricinin yöntemleri incelendiğinde yazılımdaki sınıf sayısını dikkate almadıkları görülmüştür.

Yöntem ile ilgili eleştirilerim, görüşlerim:

- Metriklerin bazıları yazılım sisteminin tamamına yönelik bazıları ise sınıflara yöneliktir.

  Sınıflara yönelik metrik değerlerinin ortalaması alınarak bütün yazılım sistemi için tek bir metrik değeri üretilmiştir.

  Ancak aralık ölçeğinden aşağıda olan ölçeklerde (örneğin sıralama ölçeği) ortalama almak anlamlı değildir.

  Uyum (*cohesion*), karmaşıklık (*complexity*) gibi özellikler aralık ölçeğinde ölçülemezler.

  Ayrıca sıra dışı değerler içeren kümlerde de ortalama yanıltıcı olur.
- Metriklerin değer aralıkları çok farklı olduğundan normalizasyon işlemi yapılıyor.

  İlk örnekte bir yazılım sisteminin incelenen ilk sürümündeki metrik değerleri 1 kabul edilip sonraki sürümlerde bu metrik değerlerinin oransal olarak nasıl (kaç kat) değiştiği hesaplanıyor.

  Bu tür bir normalizasyon yapılabilmesi için metriklerin orantı (*ratio*) ölçeğinde olması gerekir.

  Ancak yazılımların birçok özelliği (uyum, karmaşıklık gibi) ancak sıralama ölçeğinde ölçülebilir.

Yöntem ile ilgili eleştirilerim, görüşlerim (devamı):

- Normalizasyon işlemleri nedeniyle metrikler artık bağıl değerlere sahip olurlar, bu değerler kendi başlarına bir anlam taşımazlar.

  Metrik değerleri ve onları kullanılarak elde edilen sonuçlar, normalizasyon işlemine bağlı olarak belli, bir referans noktasına göre anlam taşırlar.

  Örnek 1:

  Tüm metrik değerleri projenin belli bir referans sürümündeki değerler 1 kabul edilerek normalize edilirse, kalite nitelikleri için elde edilen değerler de referans sürüme göre değişimi ifade ederler.

  Kalite niteliklerinin iyi veya kötü yönde ne kadar değiştiği belirlenebilir.

  Elde edilen değer, tek başına o projenin kalite düzeyinin yeterliği konusunda bilgi vermez.

  Örnek 2:

  Metrikler, n adet projedeki değerlere göre 1 ile n arasında normalize edilirler.

  Bu durumda kalite nitelikleri için elde edilen değerler projeler arasında sıralama yapılmasını sağlarlar.

  Değerler yine tek başına anlamlı değillerdir.

  Projenin kalite düzeyinin yeterliği konusunda karar vermek için bir referans projeye gerek vardır.