Yazılım Kalitesinin Ölçülmesi (*Software Quality Measurement*)

Yazılım projesinde kalite nitelikleri ölçülebilen varlıklar (*entity*) üç grupta toplanır.

1. **Süreçler** (*processes*): Zamana bağlı etkinliklerdir. Belli sürelerde ve uygun sıralarda bitirilmeleri gerekir.
   Örneğin; isteklerin toplanıp dokümanın oluşturulması (analiz) süreci, tasarım süreci, belli modüllerin test edilmesi süreci gibi.
2. **Ürünler** (*products*): Yazılım geliştirme süreçlerinde oluşturulan her türlü ürün.
   Örneğin; istekler dokümanı, tasarım diyagramı, kod, açıklama dokümanı gibi.
   Ürünler bazı süreçlerde oluşturulur, bazı süreçlerde ise giriş olarak kullanılırlar.
3. **Kaynaklar** (*resources*): Süreçlerdeki etkinliklerin yürütülmesi için gerekli olan her türlü kaynak. Personel, araçlar, mekan gibi.

Ölçme işlemlerinde bu varlıkların **nitelikleri** (*attribute*) ölçülür. Örnekler:
Test sürecinin (varlık) **verimi** (nitelik) (hataların ne kadarını bulabiliyor) ölçülür.
Tasarım ürününün **karmaşıklığı**, kodun **anlaşılırlığı** ölçülebilir.
Bir kodlayıcının **verimi** (hafta kaç satır kod yazıyor) ölçülebilir.

Bir yazılımla ilgilenen farklı paydaşlar (roller) (*stakeholder*) vardır; kullanıcı, kodlayıcı, test sorumlusu, müşteri gibi.

Farklı paydaşların farklı nitelikler ile ilgili beklentileri olur; fiyat, kolaylık, hız gibi.

Yazılım Kalitesinin İç ve Dış Nitelikleri

Yazılımlarla ilgili varlıkların (süreç, ürün, kaynak) nitelikleri (*attribute*) ikiye ayrılırlar: iç nitelikler ve dış nitelikler.

İç nitelikler (*internal attributes*): Sadece ilgili varlık incelenerek ölçülebilen niteliklerdir.

Üzerinde ölçüm yapılan varlığın diğer varlıklarla veya dış dünya ile etkileşime girmesine gerek yoktur.

Örneğin, bir programın satır sayısı (*Line of Code* –LOC) programın iç niteliğidir.

Programı çalıştırmadan satır sayısı ölçülebilir.

Sınıflar arası bağımlılık da (*Coupling between objects* – CBO) bir iç niteliktir.

Dış nitelikler (*external attributes*): Dış nitelikler varlıkların davranışları ile ilgilidir.

Dış niteliklerin ölçülebilmesi için üzerinde ölçüm yapılan varlığın çevresi ile etkileşime girmesi gerekir.

Örneğin bir programın ürettiği sonuçların doğruluğu, o programın çalıştığı bilgisayara ve programı kullanan kişiye de bağlıdır.

Doğruluğu ölçmek için programın çalışması gerekir.

Dış niteliklerin, iç nitelikler kullanılarak ölçülmesi (veya öngörülmesi):

Paydaşların çoğunlukla ölçmek istedikleri yazılımın dış nitelikleridir (hız, güvenilirlik, doğruluk, kolay kullanım gibi).

Ancak dış nitelikleri ölçmek aşağıdaki nedenlerden dolayı zordur.

- Yazılımın tamamlanmış (ve çalışır durumda) olması gerekir.
- Tüm olası dış koşulları (kullanıcı tipi, bilgisayar konfigürasyonu) yaratmak zordur.
- Çok uzun sürebilir. Örneğin bir programın güvenilir (*reliable*) olduğunu anlamak için aylarca çalıştırmak gerekebilir.

İç nitelikleri ölçmek daha kolaydır; yazılımın bitmesine ve değişik koşullarda çalışmasına gerek yoktur.

Dış nitelikleri etkileyen iç nitelikleler belirlenip aralarındaki ilişkiyi gösteren bir model oluşturulabilirse, sadece iç nitelikler ölçülerek (karmaşıklık, bağımlılık, metot sayısı gibi) dış niteliklerin olası değerleri hakkında bilgi oluşturulabilir (öngörü).

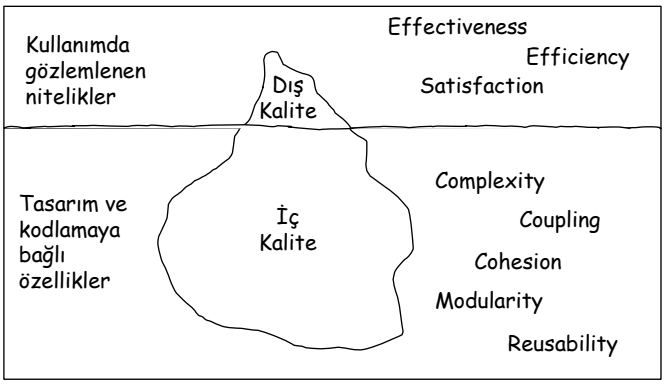
Model (İç Nitelikler) → Dış Nitelikler

Örnek:
İkinci el otomobil alacak olan müşterinin asıl ilgilendiği otomobilin dış nitelikleridir; yakıt tüketimi, sürüş güvenliği, arıza sıklığı gibi.
Ancak bunları kısa sürede ölçmek mümkün değildir.
Bu nedenle otomobilin iç niteliklerine (fren balataları, lastiklerin gibi) bakılarak dış nitelikleri ile ilgili kestirim yapılır.

Yazılım Kalitesi Buzdağı

Yazılımın dışarıdan gözlemlenen kalitesi (hız, doğruluk, hata sıklığı) büyük ölçüde tasarımının (iç yapısının) kalitesine bağlıdır.

Yazılım Kalitesi Buzdağı



Sınama (*testing*), doğrulama (*verification*), kalite ölçümü (*quality assessment*)

Yazılım geliştirme süreçlerinin içinde yer alan sınama (*testing*) ve doğrulama (*verification*) işlemleri, bazı kalite nitelikleri hakkında bilgi verirler; ancak yazılımın tüm kalitesinin ölçülmesini sağlayan işlemler değildirler.

Sınama (*test*) mantık hatalarının sıklığı konusunda fikir verir.

Doğrulama (*verification*) ise işlevsel isteklere uygunluk oranını gösterir.

Ancak bu işlemler kalite nitelikleri içinde yer alan bakım kolaylığı, tekrar kullanılabilirlik, taşınabilirlik gibi kavramların değerlendirilmesini sağlamazlar.

Örneğin bir program modülü testlerde hiç hata çıkarmamış olabilir, ancak bu modülü güncellemek istediğinizde sorunlarla karşılaşabilirsiniz.

Ayrıca sınama ve doğrulama işlemleri yazılımın belli modüllerinin kodlanıp tamamlanmış olmasını gerektirirler.

Bir yazılım sisteminin tasarım kalitesinin değerlendirme işleminin kodlamanın bitmeden projenin erken aşamalarından itibaren yapılabilmesi istenir.

Kalite ölçme ve değerlendirme yöntemleri sadece sistemin o andaki durumunu değil, ileride sorun çıkartabilecek düzeltilmesi gereken birimleri de göstermelidir.

Böylece proje devam ederken her aşamada kalitenin iyi düzeyde tutulmaya ve ileride ortaya çıkabilecek olası sorunların erken aşamalarda önlenmeye çalışılır.

Yazılım kalitesini ölçmek ve değerlendirmek için gerekli olan unsurlar:

1. Kalite modeli:

   Herhangi bir niteliği ölçüp değerlendirebilmek için bunun net bir biçimde tanımlanması (modellenmesi) gerekir.

   Ölçülmek (ve gerekiyorsa iyileştirilmek) istenen nedir?

   Yazılım kalitesini belirleyen bileşenler (alt bileşenler) nelerdir? Nasıl tanımlanırlar?

   Örneğin, bakım kolaylığı hangi bileşenlerden oluşur; hangi iç nitelikler ile modellenebilir?

2. Yazılım ölçme yöntemleri:

   Nasıl ölçülecek, veri toplanacak?

   Hangi metrikler?

3. Değerlendirme yöntemi:

   Sonuçlar nasıl yorumlanacak?

   Metriklerin sınır değerleri?

   Metrik-kalite niteliği ilişkileri?

4. Doğrulama:

   Yapılan değerlendirme geçerli mi?

   Ölçme sonuçları gerçek değerlere ne kadar yakın?

   Değerlendirme yöntemi genel mi, yazılım sisteminin özelliklerine (türü, boyu) bağlı mı?

   Gerçek dünyadaki yazılımlardan sağlıklı veri nasıl toplanacak?

Yazılım Kalite Modelleri

Neyi ölçeceğiz?

Herhangi bir şeyi (örneğin yazılım kalitesini) ölçüp değerlendirebilmek için önce ölçülecek olan kavramın net bir biçimde tanımlanması (modellenmesi) gerekir.

McCall Kalite Modeli (Factor/Criteria/Metric "**FCM**"):

Günümüzde kullanılan bir çok yazılım kalitesi modelinin temeli McCall * tarafından ABD hava kuvvetleri için 1977'de oluşturulan modele dayanmaktadır.

Yazılımın kullanıcıları ile geliştiricileri arasında anlaşmayı kolaylaştırmayı hedeflemektedir.

Yazılım kalitesi üç temel bakış açısıyla (*major perspective*) ele alınır:

- *Product revision:* Yazılımın bakımının yapılabilmesi, değişen isteklere göre güncellenebilmesi, hataların bulunabilmesi
- *Product transition*: Yazılımın yeni ortamlara (donanım) taşınabilmesi, eski modüllerin yeni yazılımlarda kullanılabilmesi, değişik birimlerle birlikte çalışabilmesi
- *Product operation*: Yazılımın kullanılması sırasındaki kalitesi ile ilgilenir (doğruluk, güvenilirlik, kolaylık gibi)

* 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.

McCall Kalite Modeli (devamı)

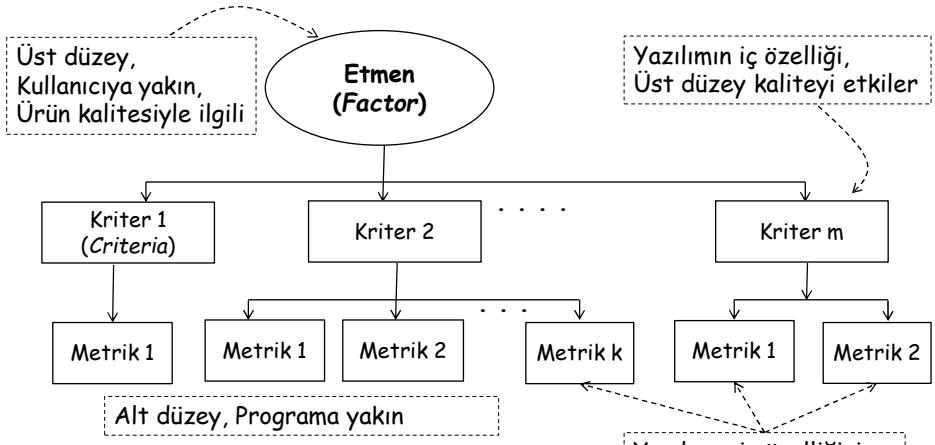
Bu katmanlı, hiyerarşik model türü Factor/Criteria/Metric (**FCM**) olarak adlandırılır.

- Etmenler (*factor*): Her bakış açısına göre yazılımın kalitesini belirleyen (sağlanması istenen) kalite etmenleri (*factors*) belirlenir.
  Etmenler, yazılımın dışarıdan (kullanıcı tarafından) görünen özellikleridir.
  Örneğin "*Product operation*" bakış açısına göre: verimlilik (*efficiency*).
- Kriterler (*criteria*): Her kalite etmenini etkileyen yazılımın iç özellikleri olan kriterler belirlenir.
  Kriterler yazılımı içeriden (geliştirici tarafından) görünen özellikleridir.
  Örneğin bellek verimliliği (*Storage efficiency*)
- Metrikler: Kriterleri ölçmek (değer atamak) için tanımlanırlar. Yazılımdan elde edilen somut değerlerdir.
  Metrikler evet/hayır yanıtı verecek şekilde hazırlanabilir.
  Bu durumda belli bir kritere bağlanan metriklerin çoğu evet yanıtı veriyorsa o kriterin sağlandığı düşünülür.
  Kriterlerde etmenlere, etmenlerden de kaliteye geçilir (aşağıdan yukarıya).

Modelin temel felsefesi, tüm etmenlerin bir araya gelmesiyle yazılım toplam kalitesini ortaya koyan bir resmin meydana geleceğidir.

McCall Kalite Modeli (devamı)

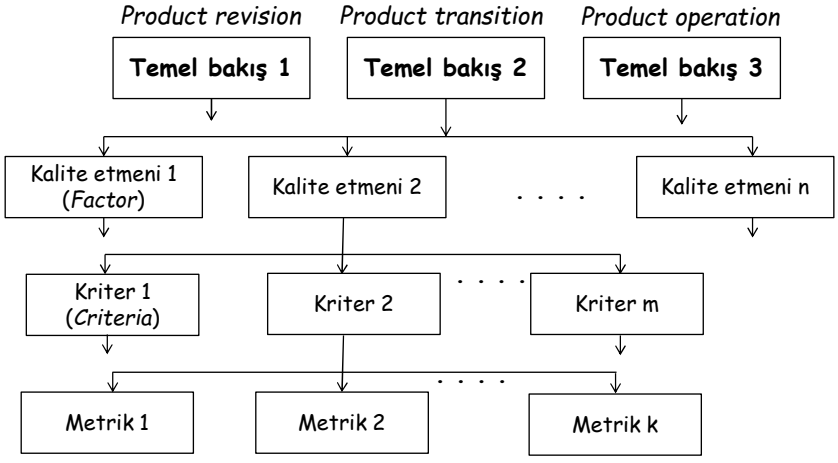
Hiyerarşik Factor/Criteria/Metric (**FCM**) modeli:



Metrik değerlerini belirleyen ölçümlerin yazılımın geliştirilmesi devam ederken yapılması amaçlanır. Bunlar proje bittikten sonra yapılan test sonuçları değildir.

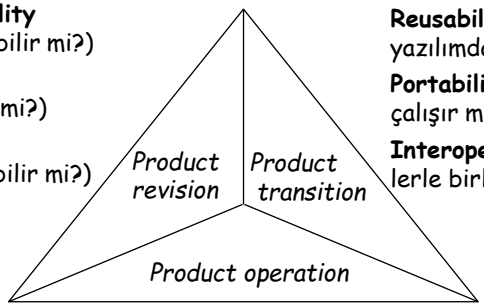
McCall Kalite Modeli (devamı)

Her bakış açısına göre farklı kalite etmenleri tanımlanmaktadır:



McCall Kalite Modeli (devamı)

Maintainability (İyileştirilebilir mi?)
Testability (Sınanabilir mi?)
Flexibility (Değiştirilebilir mi?)



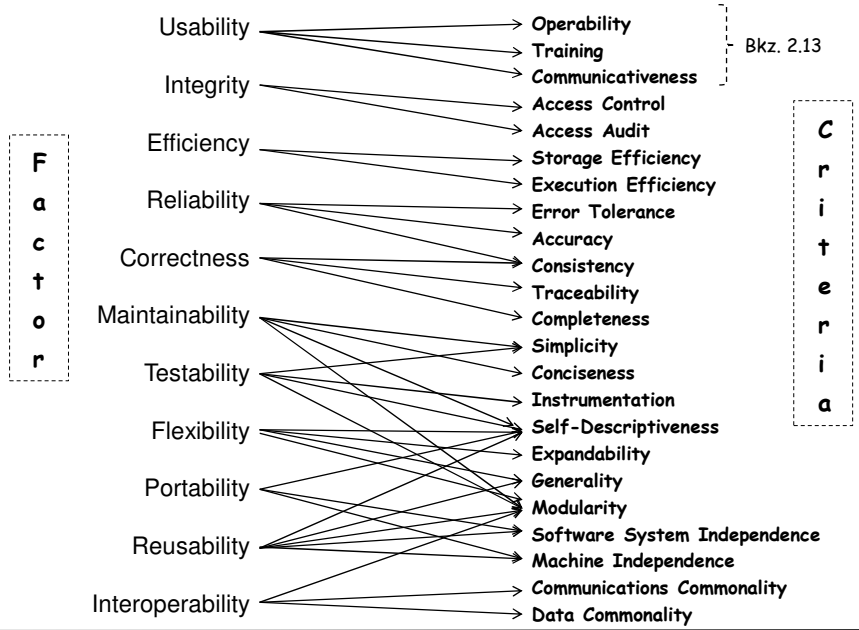
Reusability (Modülleri başka yazılımda kullanılabilir mi?)
Portability (Başka ortamda çalışır mı?)
Interoperability (Başka sistemlerle birlikte çalışabilir mi?)

Etmenler (*Factors*)

Usability (Kullanılabilir mi?)
Integrity (İstemeyen erişimlerden korunma, güvenlik)
Efficiency (Donanım üzerinde beklenen şekilde çalışacak mı?)
Correctness (İstenen işi yapıyor mu mu?)
Reliability (Sürekli doğru sonuç üretiyor mu?)

Etmenler (*Factors*)

McCall Kalite Modeli (devamı)



Örnek:

Factor:

Usability: The effort needed for use the system.

Criteria:

• **Training (Ease of learning):** The degree to which user effort required to learn how to use the software is minimized

• **Operability:** The degree to which the effort required to perform an operation is minimized

• **Communicativeness:** The degree to which software is designed in accordance with the psychological characteristics of users

Metrics:

Training:
- **Learning time:** Time for new user to learn how to perform basic functions of the software

Operability:
- **Operation time:** Time required for a user to perform operation(s) of the software

Communicativeness:
- **Human factors:** Number of negative comments from new users regarding ergonomics, human factors, etc.

Kalite etmenleri (*factor*) ile metrikler arasındaki ilişkinin kurulması:

Bilinen projeler incelenerek, deneyimlere, gözlemlere dayanarak etmenler ile metrikler arasında ilişki kurulur.

$$r_f = f(m_1, m_2, m_3, \ldots, m_n)$$

r_f : Kalite etmeni ile ilgili değerlendirme
m_i : Metrik değerleri

Örneğin regresyon analizi yapılır.

$$r_f = c_1m_1 + c_2m_2 + c_3m_3 + \ldots$$

c_i : Regresyon katsayıları

Metriklere "yok"/"var", "uygun değil"/"uygun", "0/1" gibi ikili değerler de atanarak değerlendirme için lojik fonksiyonlar da kullanılabilir.

Oluşturulan ölçme yöntemlerinin geçerliliği (doğruluğu) gerçek dünyayı ne kadar temsil ettiğine (*representation*) bağlıdır.

Ölçme sonucu elde ettiğimiz değerler ilgili yazılımın belirlemek istediğimiz kalite niteliğini ne kadar yansıtmaktadır?

Ölçmede temsil ilkesi ders notlarının 3. bölümünde "Ölçme Bilimi ile İlgili Temel Bilgiler" açıklanmıştır.

Örnek:

Factor: Correctness

Criteria: Completeness, Consistency, Traceability

Metrics:

Completeness (tamlık) değerini belirlemek için aşağıdaki ölçümler kullanılabilir. Her bir soruya karşılık Evet (1) veya Hayır (0) yanıtı verilip, ortalama alınabilir.

1. Tüm veriler tanımlı mıdır?
2. Referans verilen tüm fonksiyonlar tanımlı mıdır?
3. Tanımlanan tüm fonksiyonlar kullanılmış mıdır?
4. Tüm karar noktalarının koşulları ve dallanma sonrası işlemleri belli midir?
5. Raporlanan tüm sorunlar çözülmüş müdür?
6. Tasarım isterler ile uyumlu mudur?

Bu altı soruya verilen yanıtların toplamı 6'ya bölünerek tamlık (*completeness*) için 0 -1 arası bir değer atanabilir.

Diğer kriterler tutarlılık (*consistency*) ve izlenebilirlik (*traceability*) için de benzer şekilde değer atama yöntemleri oluşturulabilir.

Eğer üç kriterin eşit ağırlıklı olduğu kabul edilirse doğruluk (*correctness*) faktörünün değerini belirlemek için üçünün ortalaması alınabilir.

Tarif edilen bu yöntem sadece olası bir örnektir.
Gerçek dünyayı temsil edebilecek farklı yöntemler oluşturulabilir.

Hedef/Soru/Metrik Yaklaşımı
(*The Goal/Question/Metric Approach - GQM*)

Okunacak yazılar:

- Victor R. Basili, " Software Modeling and Measurement: The Goal/Question/ Metric Paradigm", Institute for Advanced Computer Studies. Department of Computer Science, University of Maryland, 1992
  http://www.cs.umd.edu/~basili/publications/technical/T78.pdf

- Victor R. Basili, Gianluigi Caldiera, H. Dieter Rombach," The Goal Question Metric Approach ", Encyclopedia of Software Engineering, Wiley, 1994. Maryland Üniversitesi, Bilgisayar Bilimleri Bölümü, ftp sunucusu
  ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf

- V. R. Basili, M. Lindvall, M. Regardie, C. Seaman, J. Heidrich, J. Munch, D. Rombach, and A. Trendowicz, "Linking Software Development and Business Strategy Through Measurement," *Computer*, vol. 43, no. 4, pp. 57–65, Apr. 2010.
  http://dx.doi.org/10.1109/MC.2010.108

Hedef/Soru/Metrik Yaklaşımı (*The Goal/Question/Metric Approach - GQM*)

Model oluşturmak için kullanılan bir yaklaşımdır.

Amaç üst düzey kalite hedefleri ile alt düzeydeki uygun metrikleri bağlamaktır.

Victor Basili tarafından NASA GSFC'deki (Goddard Space Flight Center) projelerde ortaya çıkan hataları değerlendirmek üzere oluşturulmuştur.

İlk olarak özel projeler için oluşturulsa da daha sonra kalite iyileştirme kavramına (*Quality Improvement Paradigm* –QIP) uygun olarak genişletilmiştir.

Yazılım firmalarının yaptıkları temel hatalardan biri de ya projelerinin hedeflerini açıkça belirlememek ya da belirledikleri hedeflerin anlamlarını akça ortaya koymamaktır.

Örneğin firma yazılımının güvenilir ve kolay genişletilebilir olmasını hedefler.

Ancak bu hedeflerin hangi bileşenlerden oluştuğunu ve bunların nasıl ölçüleceğini belirlemez.

Sonuçta bu hedeflere ya ulaşılamaz ya da ulaşılıp ulaşılamadığı belirlenemez.

Gilb's Principle of Fuzzy Targets: *Projects without clear goals will not achieve their goals clearly* (Tom Gilb).

GQM Yaklaşımı:

Amaç, rastgele veri toplayıp sonra bunlardan anlam çıkarmaya çalışmayı (*fishing for results*) önlemektir.

Ölçme yukarıdan (hedeflerden/gereksinimlerden) aşağıya (metriklere) olmalı (Metriklerle başlanmamalı).

1. Önce projenin/kuruluşun yazılımla ilgili hedefleri belirlenir.
2. Bu hedeflere ulaşılıp ulaşılmadığını belirleyen sorular oluşturulur.
3. Sorular incelenerek bu soruların yanıtlarını verebilecek metrikler oluşturulur.
4. Metriklerin uygulanabilirliği (veri toplanabilir mi) incelenir.

Yararı:

- Metrik kümesi küçük tutuluyor, gereksiz metriklerle zaman kaybedilmiyor.
- Toplanan veriler yararlı ve anlamlı oluyor.
- Yazılım hedefleri ile toplan veriler (metrikler) arasında anlamlı bağlantılar kuruluyor.

GQM Yapısı:

Üç katmandan oluşur:

1. Kavramsal katman (HEDEF) (*Conceptual level -GOAL*):

Hedef; bir nesne/varlık için, çeşitli amaçlarla, çeşitli kalite modellerine göre, değişik bakış açılarıyla, belli bir ortama göre tanımlanır.

2. İşlemsel katman (SORU) (*Operational level -QUESTION*):

Belli bir hedefin değerlendirilmesi veya sağlanması için izlenecek olan yolu belirleyen sorular sorulur.

Sorular, ölçmeye konu olan varlığı (ürün, süreç, kaynak) istenen kalite kriterine (süre kısaltma, maliyet düşürme vs.) ve bakış açısına göre karakterize etmelidir.

3. Nicel katman (METRİK) (*Quantitative level -METRIC*):

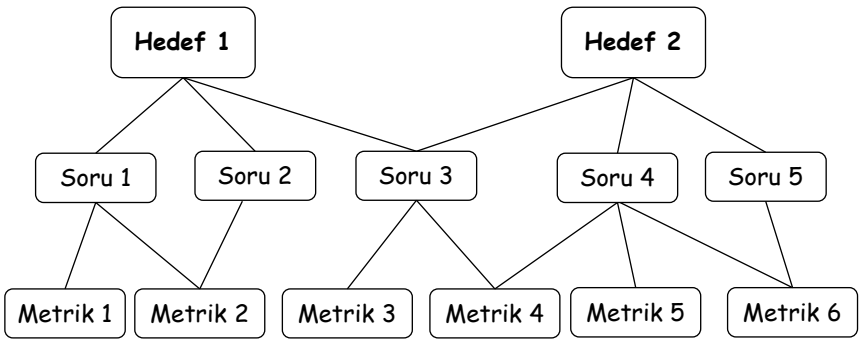
Her soruya onu nicel (sayısal) olarak yanıtlayacak veriler (metrikler) atanır.

Veriler iki tür olabilir:

- Nesnel (*objective*): Sadece ölçülen nesneye bağlıdır, bakış açısına değil.
- Öznel (*subjective*): Hem ölçülen nesneye hem de bakış açısına bağlıdır.

Örneğin müşteri memnuniyeti, kodun okunurluğu.

GQM Yapısı (devamı):



Hedef :

- Ölçmenin amacı
- Ölçülecek nesne/varlık/kavram
- Hangi bakış açısına göre ölçülecek

Soruların belirlenmesinde var olan modellerden de yararlanılabilir.

Örnek: GQM Uygulaması

Bir sistemin bakımı sırasında gelen değişiklik isteklerinin karşılanma süresinin iyileştirilmesi amaçlanıyor.

Bu örnekte beş soru ve yedi metrik ile model oluşturulmuştur.

Hedefi değişik yönlerden inceleyen sorular attırılabilir.

Goal	Purpose Issue Object (process) Viewpoint	Improve the timeliness of change request processing from the project manager's viewpoint
Question	Q1	What is the current change request processing speed?
Metrics	M1 M2 M3	Average cycle time Standard deviation % cases outside of the upper limit
Question	Q2	Is the (documented) change request process actually performed?
Metrics	M4 M5	Subjective rating by the project manager % of exceptions identified during reviews

Sonraki yansıda devam ediyor.

Örnek: GQM Uygulaması (devamı)

Question	Q3	What is the deviation of the actual change request processing time from the estimated one?
Metrics	M6	$\frac{\text{Current average cycle time - Estimated average cycle time}}{\text{Current average cycle time}} * 100$
	M7	Subjective evaluation by the project manager
Question	Q4	Is the performance of the process improving?
Metrics	M8	$\frac{\text{Current average cycle time}}{\text{Baseline average cycle time}} * 100$
Question	Q5	Is the current performance satisfactory from the viewpoint of the project manager?
Metrics	M7	Subjective evaluation by the project manager

ISO/IEC Yazılım Kalite Modeli Standartları

Neyi ölçeceğiz?

Okunacak dokümanlar:

ISO/IEC 250mn Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) Serisinden aşağıdaki dokümanlar okunacaktır:

- ISO/IEC 25000 : 2014 Guide to SQuaRE
- ISO/IEC 25010 : 2011 System and software quality models
- ISO/IEC 25020 : 2019 Measurement reference model and guide
- ISO/IEC 25021 : 2012 Quality measure elements

Standartların dokümanlarına İTÜ kampüsü içinden Biritish Standarts Online sayfasında **https://bsol.bsigroup.com**/ adresinden erişebilirsiniz.

ISO/IEC Yazılım Kalite Modeli Standartları

ISO: The International Organization for Standardization
IEC: The International Electrotechnical Commission

ISO/IEC yazılım kalitesi ile ilgili tanımları netleştirmek ve standart hale getirmek için 1991 yılında 4 bölümlük ISO/IEC 9126 " Software engineering - Product quality" dokümanını oluşturmuştur.

ISO/IEC 9126-1 Part 1: Quality Model, güncelleme 2001.

ISO/IEC 9126-2 Part 2: External Metrics, güncelleme 2003.

ISO/IEC 9126-3 Part 3: Internal Metrics, güncelleme 2003.

ISO/IEC 9126-4 Part 4: Quality in use Metrics, güncelleme 2004.

Yazılım kalitesinin nasıl ölçülüp değerlendirileceğini belirten kısımlar 9126:1991 standardından çıkartılarak 1999 yılında ISO/IEC 14598 "Information technology — Software product evaluation" adıyla yayımlandı.

Bu standartlar yeniden ele alınıp güncellenerek 2005'ten itibaren ISO/IEC 250XX "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE)" adıyla yayımlanmaya başladı.

ISO/IEC 9126-1'in karşılığı 2011'de ISO/IEC 25010 "System and software quality models" olarak yayımlandı.

Yazılım Kalite Modeli Standartlarının Amacı

- Yazılım kalitesi ile ilgili bileşenleri ve aralarındaki ilişkileri tanımlamak
  - Bir yazılımın kalitesini belirleyen unsurlar nelerdir?
- Yazılım kalitesi konusunda çalışan kişiler arasında **ortak bir dil** oluşturmak
  - Yazılım kalitesi ile ilgili terimlerin tanımını net biçimde yapmak
  - Bir terimin herkes için aynı anlama gelmesini sağlamak
- Standartlarda olmayanlar:
  - Kalite ölçümü (öngörü) için verilerin hangi yöntemlerle toplanacağı
  - Ölçmede (öngörüde) kullanılacak olan parametreler ve yöntemlerin ayrıntıları

  Özellikle sayısal olarak ifade edilmesi zor olan kalite bileşenlerinin ölçülmesi için yöntemlerin geliştirilmesi konusunda çalışmalar devam etmektedir.

  Sayısallaştırılması zor olan bileşenlere örnekler; öğrenilebilirlik (*learnability*), değiştirilebilirlik (*modifiability*), tekrar kullanılabilirlik (*reusability*)

Yazılım Kalite Modeli Standartlarının Kullanım Yerleri

1. Yazılımı geliştirmeden önce kalite isteklerini (beklentilerini) belirlemek
2. Yazılım geliştirdikten sonra kaliteyi ölçüp isteklerle karşılaştırmak

ISO/IEC 250mn

**Systems and software engineering —
Systems and software Quality Requirements and Evaluation (SQuaRE) —
Standartları**

ISO/IEC 9126 ve ISO/IEC 14598 standartlarının yerine gelmiştir.

Bölümler:

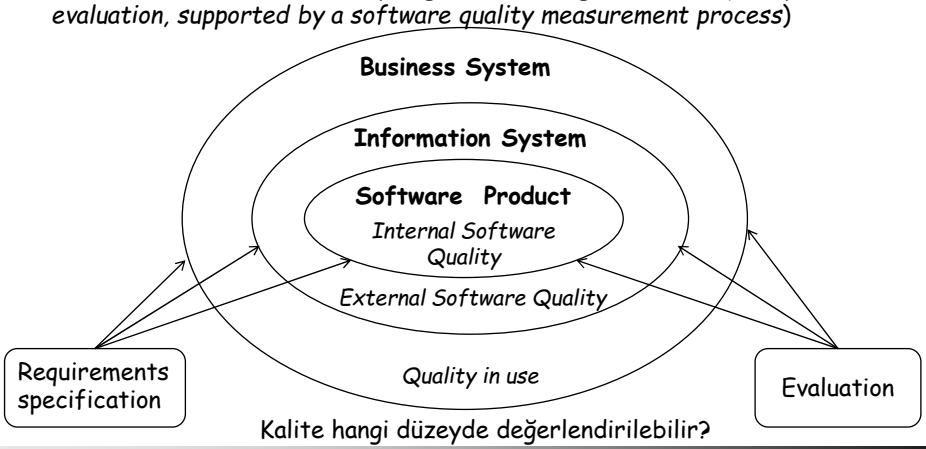
- ISO/IEC 2500n - Quality Management Division: Genel bilgiler ve tanımlar
- ISO/IEC 2501n - Quality Model Division: İç, dış, kullanımdaki kalite karakteristiklerini ve veri kalitesi modellerini açıklar.
- ISO/IEC 2502n - Quality Measurement Division: Kalitenin ölçülmesi ile ilgili matematiksel tanımlar
- ISO/IEC 2503n - Quality Requirements Division: Yazılımın ilgililerinden kaynaklanan kalite isteklerinin nasıl belirlenebileceğini açıklar.
- ISO/IEC 2504n - Quality Evaluation Division: Yazılımın kalitesinin nasıl değerlendirilebileceğini açıklar.

ISO/IEC 25050 - ISO/IEC 25099 arası da SQuaRE standartları ve teknik raporları için rezerve edilmiştir.

ISO/IEC 25000 Genel Bilgiler

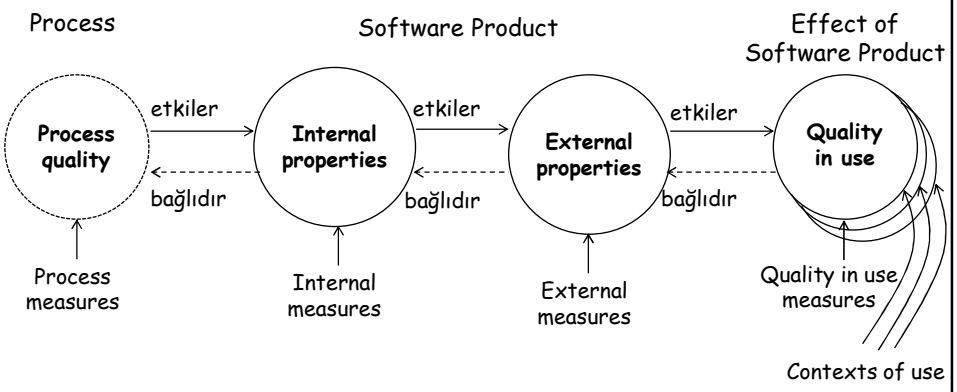
Amaç iki konuda yol göstermek:

1. İsterlerin (gereklerin) nasıl belirleneceği (*software quality requirements specification*)
2. Yazılım kalitesinin nasıl ölçülüp değerlendirileceği (*software quality evaluation, supported by a software quality measurement process*)



Kalite hangi düzeyde değerlendirilebilir?

Yazılımda kalite bileşenleri arasındaki etkileşim



İç kalite nitelikleri (*Internal quality attributes*): Yazılımın iç yapısı (tasarım, kod) ile ilgili program çalışmadan (statik) ölçülebilen nitelikler.

Dış kalite nitelikleri (*External quality attributes*): Yazılımın çalışırken **testlerle** dışarıdan ölçülen nitelikler. Yazılımın çalıştığı sisteme (ortama) bağımlıdır.

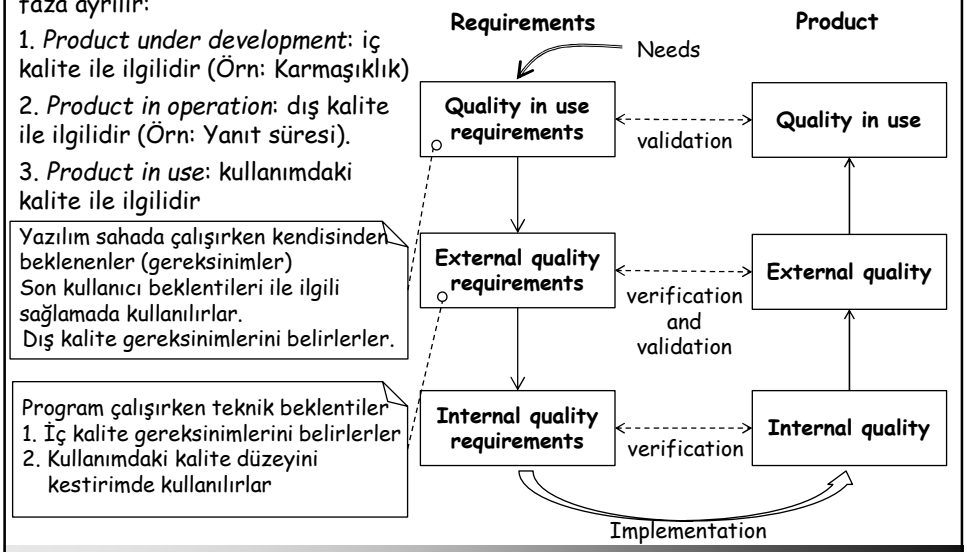
Kullanımdaki kalite nitelikleri (*Quality in use attributes*): Yazılım sahada kullanılırken kullanıcıya (geniş anlamda ilgiliye) yansıyan nitelikler.

ISO/IEC 25000 Yazılımda ürün kalitesi yaşam döngüsünün modeli

Yazılımın kalite yaşam döngüsü üç faza ayrılır:

(*Software Product Quality Life Cycle Model*)

1. *Product under development*: iç kalite ile ilgilidir (Örn: Karmaşıklık)
2. *Product in operation*: dış kalite ile ilgilidir (Örn: Yanıt süresi).
3. *Product in use*: kullanımdaki kalite ile ilgilidir



ISO/IEC 25010 : 2011

Systems and software engineering —
Systems and software Quality Requirements and Evaluation (SQuaRE) —
System and software quality models

Bu dokümanda yazılım ürünleri ve bilgisayar sistemleri ile ilgili kalite kriterleri ele alınmaktadır.

Bir sistemin **kalite**si, o sistemin ilgililerinin (*stakeholders*) (yatırımcı, proje lideri, tasarımcı, kullanıcı, müşteri, yazılımı kullanan firmanın yaptığı işten etkilenen kişiler vb.) gereksinimlerinin karşılanma miktarıdır.

Buradaki gereksinimler hem belirtilmiş olanlar (*stated*) hem de doğal olarak olması beklenenlerdir (*implied*).

Kalite modelleri bu gereksinimleri, kalite karakteristikleri ve alt karakteristikler şeklinde ortaya koyarlar.

Bu modellerin temel amaçları yol göstermek, ortak terimler oluşturmak ve unutulmaları önlemektir.

Dokümanda yer alan kalite karakteristikleri genel amaçlı olarak hazırlandıkları için hepsi her proje için gerekli olmayabilirler.

Bu nedenle kalite modelleri, üzerinde çalışılacak sisteme göre şekillendirilmelidirler.

ISO/IEC 25010 : 2011 iki kalite modeli içeriyor:

A. Kullanımdaki kalite modeli (*quality in use model*): 5 adet ana kalite karakteristiği
B. Ürün kalite modeli (*product quality model*): 9126-1:2001'deki iç ve dış karakteristikler birleştirilmiştir. 8 adet ana kalite karakteristiği ve onların alt karakteristikleri tanımlanmıştır.

Ayrıca ISO/IEC 25012'de veri kalite modeli (*data quality model*) tanımlanmıştır.

Kalite modellerinin kullanım yerleri:

- Yazılım ve sistem gereksinimlerinin belirlenmesi
- Gereksinim tanımlarının anlaşılırlıklarının onaylanması
- Yazılım ve sistem tasarım hedeflerinin belirlenmesi
- Yazılım ve sistem test hedeflerinin belirlenmesi
- Kalite kontrol kriterlerinin belirlenmesi
- Yazılım ve yazılım bağımlı sistemlerin kabul kriterlerinin belirlenmesi
- Kalite karakteristikleri ile ilgili ölçümler (metrikler) oluşturmak

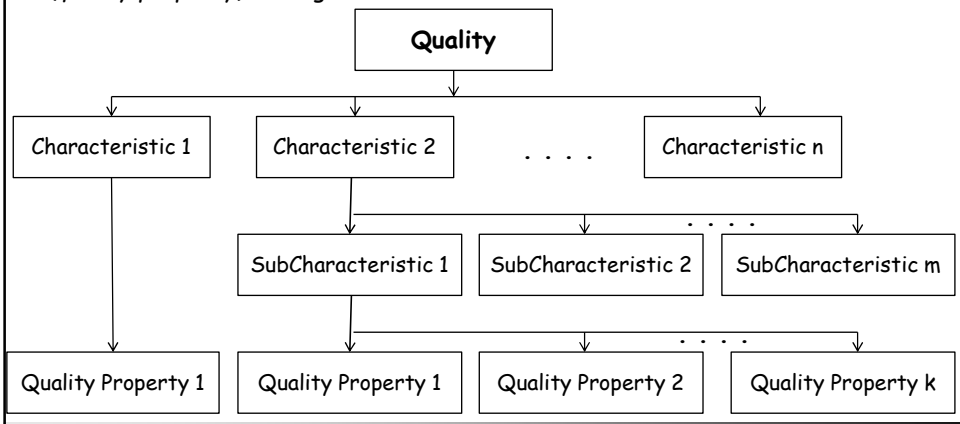
ISO/IEC 25010 : 2011 Kalite Modelleri

Kalite modelleri hiyerarşik yapıya sahiptir.

Kaliteyi ana karakteristikler belirler.

Bazı ana karakteristikler alt karakteristiklerden oluşur.

Karakteristikler (varsa alt karakteristikler) ölçülebilen kalite özellikleri (*quality property*) ile değerlendirilirler.



A. Kullanımdaki kalite modeli (*quality in use model*):

Karakteristiklerin anlamları ilgiliye (paydaş) (*stakeholder*) göre (birincil kullanıcı, bakım sorumlusu gibi) değişir.

1. Effectiveness (Etkinlik): Kullanıcı (paydaş) tipine göre belirlenen gereksinimleri sağlayan doğruluk (*accuracy*) ve eksiksizlik (*completeness*) özelliği

2. Efficiency (Verimlilik): Kullanıcının gereksinimlerini sağlayan doğruluk ve eksiksizliğe ulaşmak için harcanan kaynak miktarı

3. Satisfaction
- Usefulness
- Trust
- Pleasure
- Comfort

4. Freedom from risk
- Economic risk mitigation
- Health and safety risk mitigation
- Environmental risk mitigation

5. Context coverage
- Context completeness
- Flexibility

Karakteristiklerin anlamlarını ISO/IEC 25010 : 2011 dokümanında okuyunuz.

B. Ürün kalite modeli (*product quality model*):

1. Functional suitability (İşlevsel uygunluk): Ürün ya da sistem belirtilen koşullarda kullanıldığında, belirtilmiş olan (*stated*) ve doğal olarak kapsanan (*implied*) gereksinimleri ne oranda karşıladığının derecesidir.
- Functional completeness (işlevsel tamlık): Fonksiyonların tüm beklenen işlevleri kapsama derecesi.
- Functional correctness (işlevsel doğruluk): Fonksiyonların gerek duyulan hassasiyet ölçüsünde ne kadar doğru sonuç ürettiğinin derecesi
- Functional appropriateness: Fonksiyonların gerekli işlerin yapılmasını ne kadar kolaylaştırdığının derecesi

2. Performance efficiency
- Time behavior
- Resource utilization
- Capacity

3. Compatibility
- Co-existence
- Interoperability

B. Ürün kalite modeli (*product quality model*) (devamı):

4. Usability
- Appropriateness recognizability
- Learnability
- Operability
- User error protection
- User interface aesthetics
- Accessibility

5. Reliability
- Maturity
- Availability
- Fault tolerance
- Recoverability

B. Ürün kalite modeli (*product quality model*) (devamı):

6. Security
- Confidentiality
- Integrity
- Non-repudiation
- Accountability
- Authenticity

7. Maintainability
- Modularity
- Reusability
- Analysability
- Modifiability
- Testability

8. Portability
- Adaptability
- Installability
- Replaceability

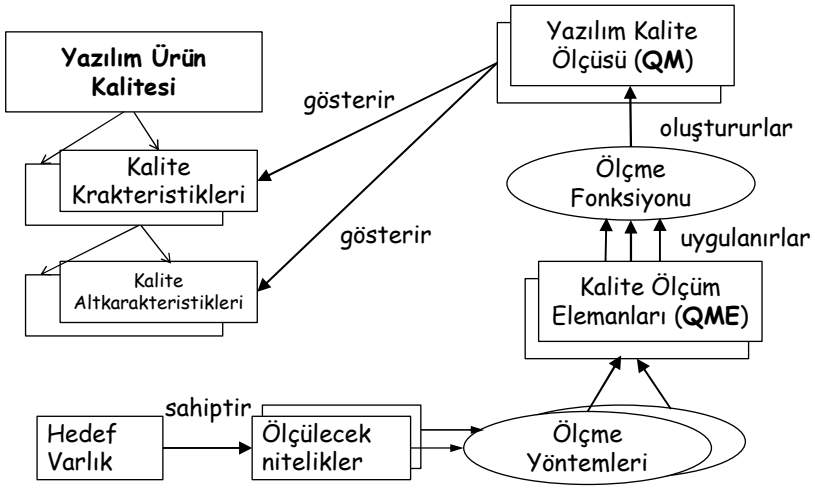
Bütün kalite karakteristiklerini her proje için belirlemek ve ölçmek pratik açıdan gerekli veya mümkün olmayabilir.

Projenin hedeflerine göre karakteristiklerin önem dereceleri belirlenmeli ve model projeye göre uyarlanmalıdır.

ISO/IEC 25020 : 2019 Measurement reference model and guide

Software quality measurement reference model (SPQM_RM):

Kalite, karakteristikler , kalite özellikleri (nitelikleri) ve ölçme işlemleri arasındaki ilişkiyi gösterir.



SPQM_RM'de ölçme zinciri

- **Kalite özellikleri** (*quality properties, property to quantify*) yazılımın nicelik olarak (örneğin sayı ile) ifade edilebilen ve kalite ölçümünde kullanılan basit özellikleridir.

  Örneğin; bir programın boyu (*size*), bir sınıfın boyu, bir sınıfın bağımlılığı. Bu özelliklere bir ölçme yöntemi ile sayısal değer atanır.

- **Ölçme yöntemi** (*measurement method*) kalite niteliklerine belli bir ölçeğe (*scale*) göre değerler atamak için uygulanan mantıksal işlemler zinciridir.

  Ölçme yöntemi; sayma, süre tutma, gözlemleme olabilir.

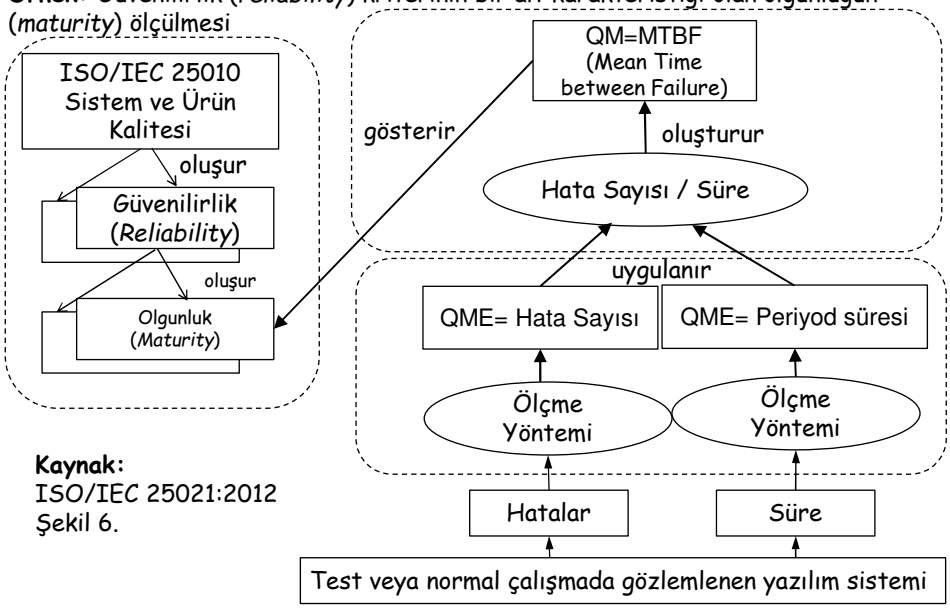
- Bir ölçme yönteminin uygulanması sonucu elde edilen değere **kalite ölçüm elemanı** (*quality measure element – QME*) denir.

- **Ölçme fonksiyonu** (*measurement function)* kalite ölçüm elemanlarını uygun şekilde harmanlayan bir algoritmadır.

- Ölçme fonksiyonunun ölçüm elemanlarına uygulanması sonucu belli bir kalite karakteristiğinin (ya da alt karakteristiğinin) değeri olan **yazılım kalitesi ölçüsü** (*software quality measure - QM*) elde edilmiş olur.

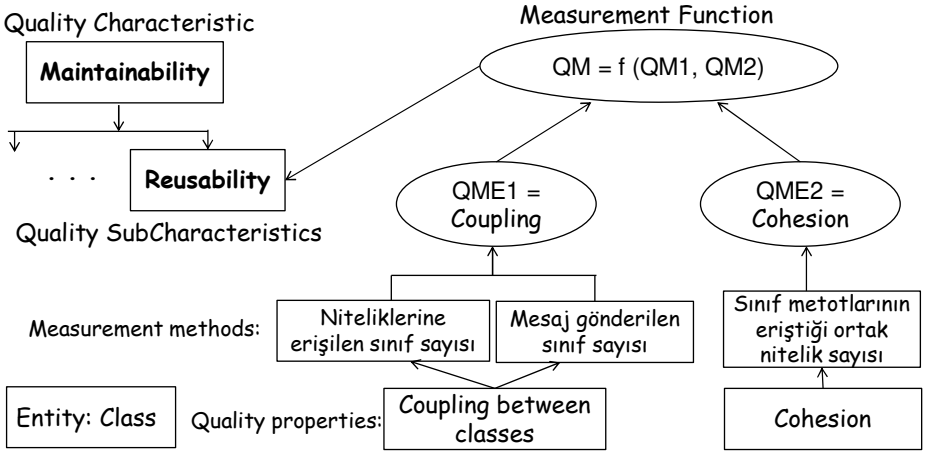
- Karakteristiklerin değerlerinin uygun şekilde birleştirilmesi sonucu ilgili kalite kriterine (*quality*) değer atanmış (yani ölçülmüş) olur.

Örnek: Güvenilirlik (*reliability*) kriterinin bir alt karakteristiği olan olgunluğun (*maturity*) ölçülmesi

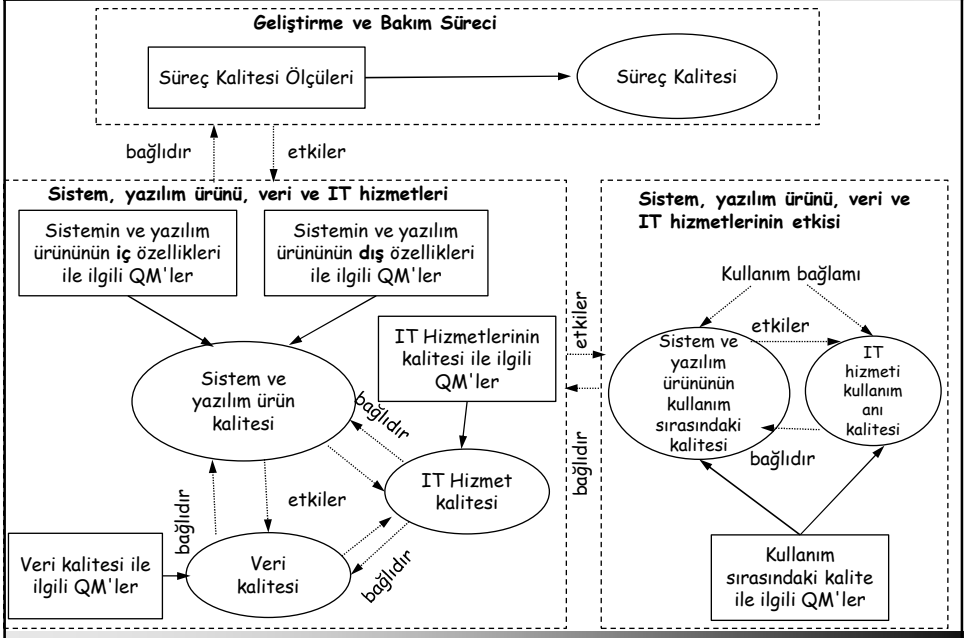


Kaynak:
ISO/IEC 25021:2012
Şekil 6.

Örnek:
Bu örnekte, bir sınıfın bakım kolaylığı (*maintainability*) kalite karakteristiğinin, tekrar kullanılabilirlik (*reusability*) alt karakteristiğinin öngörülebilmesi (*prediction*) için, SPQ-RM modelinin nasıl uygulanabileceği basit olarak gösterilmiştir.
Örnekte amaç sadece modelin uygulamasını göstermektir. **Model tam değildir.**



Farklı QM'ler arasındaki ilişkiler ISO/IEC 25020:2019



Yazılımın kalite yaşam döngüsü (*Software Quality Life Cycle*)