Yazılım Kalitesinin Ölçülmesi (*Software Quality Measurement*)

Temel Bilgiler:

Bir yazılım projesinde nelerin kalitesi ölçülür?

- Yazılım projesinde **kalite nitelikleri** (*quality attributes*) 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, test süreci gibi.
   Test sürecinin (varlık) **verimi** (nitelik) (hataların ne kadarı, ne kadar kaynak harcanarak bulunabiliyor) ölçülür
2. **Ürünler** (*products*): Yazılım geliştirme süreçlerinde oluşturulan her türlü ürün.
   Örneğin; istekler dokümanı, tasarım diyagramı, kod, açıklama dokümanı gibi.
   Tasarım modelinin (*design diagram*) **karmaşıklığı**, kodun **anlaşılırlığı** ölçülebilir.
   Bir yazılım sisteminin ne kadar kolay (veya zor) güncellenebildiği (*modifiability*) ölçülebilir.
3. **Kaynaklar** (*resources*): Süreçlerdeki etkinliklerin yürütülmesi için gerekli olan her türlü kaynak. Personel, araçlar, mekan gibi.
   Bir kodlayıcının **üretkenliği** (hafta kaç satır kod yazıyor) ölçülebilir.

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

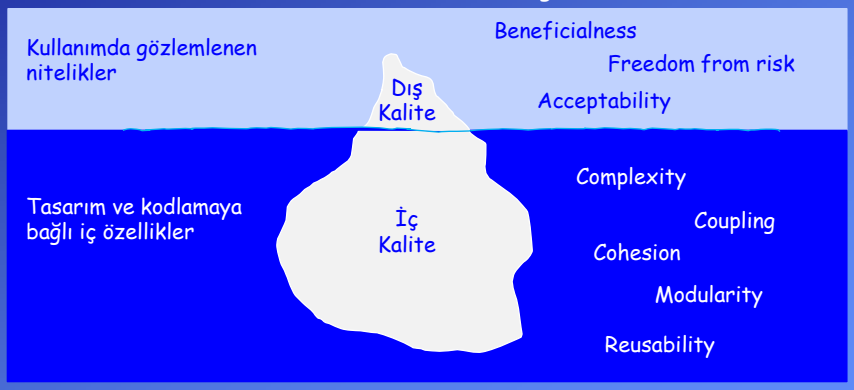
- **İç nitelikler** (*internal attributes*):
  - Sadece ilgili varlık incelenerek ölçülebilen niteliklerdir.
  - Üzerinde ölçüm yapılan varlığın diğer varlıklarla veya dış dünya ile etkileşime girmesine gerek yoktur.
  - Örneğin, bir programın satır sayısı (*Line of Code* – LOC) programın iç niteliğidir.
    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.

Yazılım Kalitesinde Buzdağı Benzetmesi

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

Yazılım Kalitesi Buzdağı



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

- Paydaşların çoğunlukla hedef koymak ve ölçmek istedikleri yazılımın dış nitelikleridir (hız, güvenilirlik, doğruluk, kolay kullanım, sürdürülebilirlik gibi).
- Ancak dış nitelikleri ölçmek aşağıdaki nedenlerden dolayı zordur (zaman ve efor gerektirir).
  - 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.
- Yazılım tamamlandıktan sora belirlenen kusurları düzeltmenin maliyeti yüksektir.
- Kusurlu birimler sisteme girmeden önce onları belirleyebilmek amaçlanır.
- 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 - *prediction*).

  Model (İç Nitelikler) → Dış Nitelikler
- Bu tür öngörü yazılım geliştirmenin erken aşamalarından itibaren yapılabilir.

Dış niteliklerin, iç nitelikler kullanılarak öngörülmesi (devamı):

Örnek 1:

- 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, kaç kilometre yol yaptığı, lastiklerin durumu gibi) bakılarak dış nitelikleri ile ilgili kestirim yapılır.

Örnek 2:

- Yazılım projelerindeki en büyük maliyet kalemlerinden biri sürdürülebilirlik (bakım) maliyeti olmaktadır.
- Ancak bu maliyetleri kısa sürede ve doğrudan ölçmek mümkün değildir.

  Bakım maliyetini doğrudan belirlemek için yazılımın tamamlanması ve sahada bir süre kullanılması, çıkan sorunların çözümlenerek raporlanması gerekir.
- Ayrıca yazılım sahaya çıktıktan sonra belirlenen tasarım kusurlarını düzeltmenin maliyeti de çok yüksek olacaktır.
- Bu nedenle, henüz yazılım geliştirilirken, yazılımın iç nitelikleri (metotlardaki karmaşıklık, sınıflar arası bağımlılık, kod satır sayısı gibi) değerlendirilerek dış nitelikleri (örneğin sürdürülebilirlik) ile ilgili öngörüde bulunulur ve önlem alınır.

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

- **Sınama** (*test*) mantık hatalarının sıklığı konusunda fikir verir.
- **Doğrulama** (*verification*) ise işlevsel isteklere uygunluk oranını gösterir.
- Sınama (*testing*) ve doğrulama (*verification*) işlemleri, bazı kalite nitelikleri hakkında bilgi verirler; ancak yazılımın tüm kalitesinin ölçülmesini sağlayan işlemler değildirler.
  - Örneğin, bu işlemler kalite nitelikleri içinde yer alan bakım kolaylığı, tekrar kullanılabilirlik, taşınabilirlik gibi kavramların değerlendirilmesini sağlamazlar.
  - Bir program modülü testlerde hiç hata çıkarmamış olabilir, ancak bu modülü güncellemek istediğinizde sorunlarla karşılaşabilirsiniz.
- Sınama ve doğrulama işlemleri yazılımın belli modüllerinin kodlanıp tamamlanmış olmasını gerektirirler.
  - Bir yazılım sisteminin kalitesinin değerlendirilmesi, kodlama bitmeden, projenin erken aşamalarından itibaren yapılabilmedir.
  - 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 nitelikle ilgili hedefler koymak ve bu 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 referans değerleri?
   - Farklı aralıklarda değer alan metrikler birlikte nasıl kullanılacak
   - 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 (nasıl) ölçeceğiz?

- Bir yazılımın kalitesi ile ilgili başlangıçta hedefler koymak ve yazılımın birimleri tamamlandıkça kaliteyi ölçüp değerlendirebilmek için önce ölçülecek olan kavramın net bir biçimde tanımlanması gerekir.
- Eğer bir dış nitelik ile ilgili öngörü yapılacaksa bu dış niteliği etkileyen iç niteliklerden oluşan bir modelin oluşturulması gerekir. **Model (İç Nitelikler) → Dış Nitelikler**

Kalite modeli:

- Günümüzde yazılım kalite modelleri genelde **hiyerarşik katmanlı** yapıda oluşturulurlar.

  Yazılımın alt düzey (koda yakın) özelliklerinden elde edilen değerler ile daha üst düzeydeki (paydaşlara / kullanıcılara yakın) karakteristiklere ilişkin elde edilmeye çalışır.

  Bu yöntem daha çok dolaylı ölçmeye yakın olup yazılım ürünü tamamlanmadan önce ilgili kalite karakteristiği ile ilgili öngörü (*prediction*) yapılmış olur.
- Kalite modelleri, nitelikleri ölçülüp değerlendirilecek olan hedef varlığın kalite karakteristiklerini (niteliklerini), alt karakteristiklerini ve aralarındaki ilişkileri tanımlarlar.
- Ölçmeyi gerçekleştirebilmek için bu karakteristikler, yazılımın nicel değerlere dönüştürülebilen özelliklerine bağlanırlar.
- Yazılımın özellikleri nicel (çoğunlukla sayısal) değerlere dönüştürülür ve elde edilen değerlerden ilgili kalite karakteristiklerinin düzeylerinin belirlenmesin sağlayan ölçme yöntemleri geliştirilir.

Örnek Yazılım Kalite Modelleri:

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

- Günümüzde kullanılan bir çok katmanlı 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)
- Her perspektifte ölçülmek istenen dış nitelikler (*factor*), onları etkileyen kriterler ve kriterleri sayısallaştırmak için kullanılacak olan metriklerin yer aldığı üç katmanlı modeller oluşturulmuştur.

Boehm Kalite Modeli:

- 1978 yılında Barry Boehm de, üst düzey dış kalite niteliklerini alt düzey iç niteliklere bağlayan katmanlı bir kalite modeli önermiştir. Barry W. Boehm, "Characteristics of Software Quality", North-Holland Pub. Comp., 1978.

* 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)

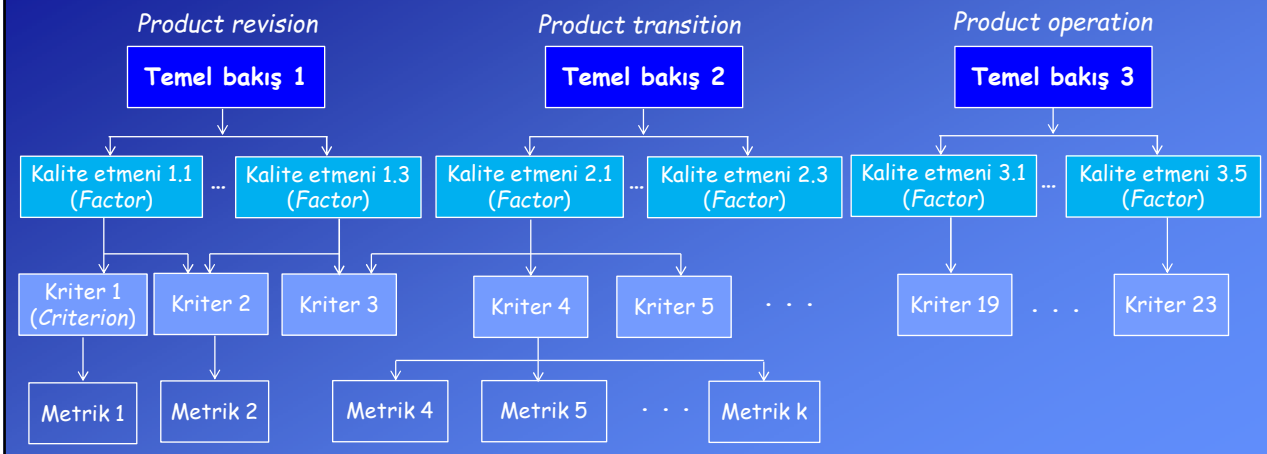
Perspektifler ve etmenler (*factors*) :

McCall Kalite Modeli (devamı)

- Bu katmanlı, hiyerarşik model Factor/Criteria/Metric (**FCM**) olarak da adlandırılır.
- Daha soyut olan dış niteliklerden (etmen) daha somut olan iç niteliklere (kriter ve metrik) doğru gidilir.
  - Etmenler (*factor*): Her bakış açısına göre yazılımın kalitesini belirleyen (sağlanması istenen) kalite etmenleri 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).

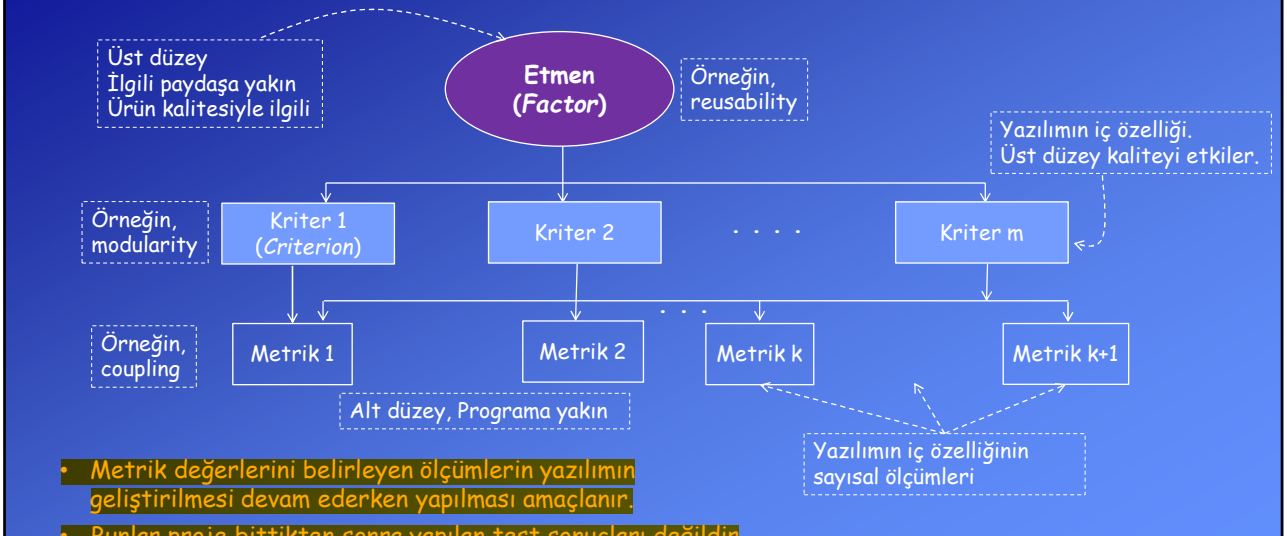
McCall Kalite Modeli (devamı):

- Her bakış açısına göre farklı kalite etmenleri tanımlanmaktadır:



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)

Örnek: *Usability in Product Operation Perspective*

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:

- **For Training, Learning time:** Time for a new user to learn how to perform basic functions of the software
- **For Operability, Operation time:** Time required for a user to perform operation(s) of the software
- **For Communicativeness, Human factors:** Number of negative comments from new users regarding ergonomics, human factors, etc.

Alt düzey metrik değerleri ile üst düzey kalite nitelikleri arasında ölçme/değerlendirme ilişkisinin kurulması:

- Modeller daha soyut olan üst düzeyden (dışarıdan) daha somut olan alt düzeye doğru (iç nitelikler) ilişkileri tanımlarlar.
- Ölçme işleminde ise alt katmanda yer alan metriklerin değerleri belirlenir ve bu değerlerden hareket ederek üst düzeydeki nitelikler değerlendirilir.
- Paydaşların o nitelikle ilgili beklentisi dikkate alınarak, bilinen projeler incelenerek, deneyimlere, gözlemlere dayanarak etmenler ile metrikler arasında ilişki kurulur.

Örnek değerlendirme yöntemleri:

- Metriklerin ayrı ayrı değerlendirilmesi:
  - Metrik değerleri incelenerek yapılması gerekenlere karar verilir.
    Örneğin öğrenme süresi uzunsa (karar için bir referans değeri gereklidir) bu konuda iyileştirme yapılır.
- Bir lojik fonksiyon ile metriklerin birlikte değerlendirilmesi:
  - Metrik değerleri belli referans değerleri ile karşılaştırılarak bir lojik fonksiyon oluşturulur.
    Örneğin; M1 > 150 ve M2 < 0.3 → Bu nitelikte iyileştirme yapılmalıdır.

Örnek değerlendirme yöntemleri (devamı):

- Metrik değerlerinin doğrudan veya normalize edilerek kullanıldığı bir regrasyon fonksiyonu oluşturulması:

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

r_f : Kalite niteliği ile ilgili değerlendirme

m_i : Metrik değerleri

Örneğin regresyon analizi yapılır.

$r_f = c_1 \times nm_1 + c_2 \times nm_2 + c_3 \times nm_3 + \ldots$

c_i : Regresyon katsayıları

nm_i : Normalize edilmiş metrik değerleri

Elde edilen değere göre kararlar verilir.

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

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?

Örnek bir değerlendirme yöntemi:

- Bu üç soruya verilen yanıtların toplamı 3'e 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.

Diğer bir örnek yöntem:

- Tamlığın kabul edilebilir olması için tüm soruların (veya çoğunluğun) yanıtlarının "evet" olması istenir.

Gerçek dünyayı temsil edebilecek farklı yöntemler oluşturulabilir.

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

- Katmanlı **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.

Kaynaklar:

- 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.
- 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.

GQM Yaklaşımı:

- Yazılım firmalarının yaptıkları temel hatalardan biri de ya projelerinin hedeflerini açıkça belirlememek ya da belirledikleri hedeflerin anlamlarını açıkça 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)

Amaç, rastgele veri (metrik) 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.

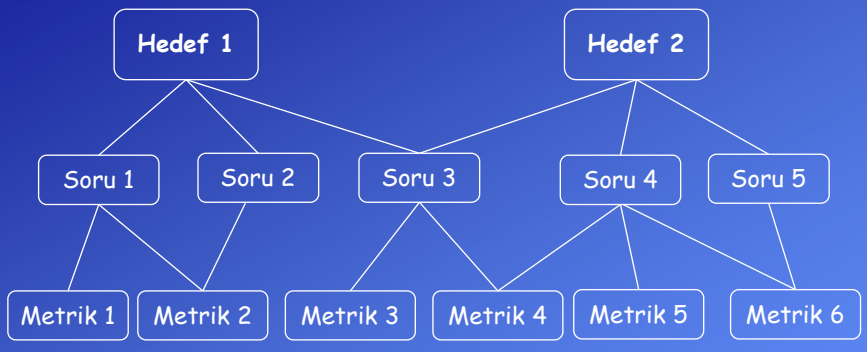
GQM Yapısı:

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 yazılım sisteminde bakım sorunları vardır. Hataların sayısı artmakta, sistemi güncellemek güçleşmektedir.

Sistemin sürülebilirliğinin iyileştirilmesi amaçlanıyor.

1. **Goal** (The "What" and "Why")

Purpose: Improve the quality and maintainability of the software product.

Issue: High number of post-release defects and difficulty for new developers to understand and modify the codebase.

Object of Study: The source code and the development process that produces it.

Viewpoint: From the perspective of the lead developer and the development team.

Context: The product team, using a Git workflow, conducts peer code reviews and runs a continuous integration (CI) pipeline.

Formal Goal Statement:

Improve the structural quality and reduce the defect density of the products codebase from the perspective of the development team, to decrease production bug-fix workload, reduce technical debt, and enhance developer productivity and morale.

Örnek: GQM Uygulaması (devamı)

2. Questions (What we need to know to achieve the goal)

Q1: How bug-prone is our current codebase? How many defects are we introducing and finding after release?

Q2: Is our code becoming more complex and difficult to maintain over time?

Q3: How effective are our code reviews at catching potential issues before they are merged?

3. **Metrics** (The data we will collect to answer the questions)

Question	Metric(s)	Data Source & Collection Method
Q1: How bug-prone?	**M1.1: Defect Density:** Number of Valid Bugs Found / Size of Codebase (e.g., in KLOC) **M1.2: Escape Rate:** Number of Bugs Found in Production / Total Bugs Found	**Jira/Bug Tracker & Git.** Count bugs by version/release. Use cloc or similar to measure code size.
Q2: Is code complex?	**M2.1: Average Cyclomatic Complexity** per function/method. **M2.2: Code Duplication Percentage.** **M2.3: Trend of these metrics over the last 6 releases.**	**Static Analysis Tools.** Integrated into CI/CD (e.g., SonarQube, Checkstyle, PMD). Reports generated on every build.
Q3: Are reviews effective?	**M3.1: Average Review Comments** per merge request. (Too low might mean rubber-stamping, too high might mean unclear code). **M3.2: Review Cycle Time:** Time from MR creation to approval. **M3.3: Defects linked to code that was reviewed.**	GitLab/GitHub/Gerrit. Use API to extract review data. Correlate bug tracker data with git blame.

Metrik değerleri incelenerek yapılması gerekenlere karar verilir.

ISO/IEC Yazılım Kalite Modeli Standartları

Neyi (nasıl) ölçeceğiz?

Kaynaklar:

ISO/IEC 250mn Systems and software engineering –
Systems and Software Quality Requirements and Evaluation (SQuaRE)

- ISO/IEC 25000: 2014 Guide to SQuaRE
- ISO/IEC 25002: 2024 Quality model overview and usage
- ISO/IEC 25010: 2023 Product quality model
- ISO/IEC 25019: 2023 Quality in-use model
- ISO/IEC 25020: 2019 Measurement reference model and guide
- ISO/IEC 25021: 2012 Quality measure elements

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

Yazılım Kalite Modeli Standartlarının Amacı

- 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
- Yazılım kalitesi ile ilgili bileşenleri ve aralarındaki ilişkileri tanımlamak
  - Bir yazılımın kalitesini belirleyen ana unsurlar ve onların alt maddeleri nelerdir?
- Standartlarda olmayanlar:
  - Kalite ölçümü (öngörü) için verilerin hangi yöntemlerle toplanacağı
  - Ölçmede (öngörüde) kullanılacak olan parametreler (alt düzeydeki metrikler) ve ölçme yöntemleri

Yazılım Kalite Modeli Standartlarının Kullanım Yerleri

1. Yazılımı geliştirmeden önce kalite isteklerini (beklentileri, hedefleri) belirlemek
2. Yazılım geliştirilirken kaliteyi ölçüp isteklerle karşılaştırmak

ISO/IEC 250mn
Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Standartları

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 – 25099 arası: Extension Division: Belli uygulama alanlarına özel standartlar ve teknik raporları için rezerve edilmiştir.

  Örneğin 2024 yılında yapay zeka sistemleri ile ilgili olarak ISO/IEC 25058 ve ISO/IEC 25059 dokümanları yayımlanmıştır.

ISO/IEC 25000: 2014 Guide to SQuaRE (Genel Bilgiler)
ISO/IEC 25002:2024 Quality model overview and usage

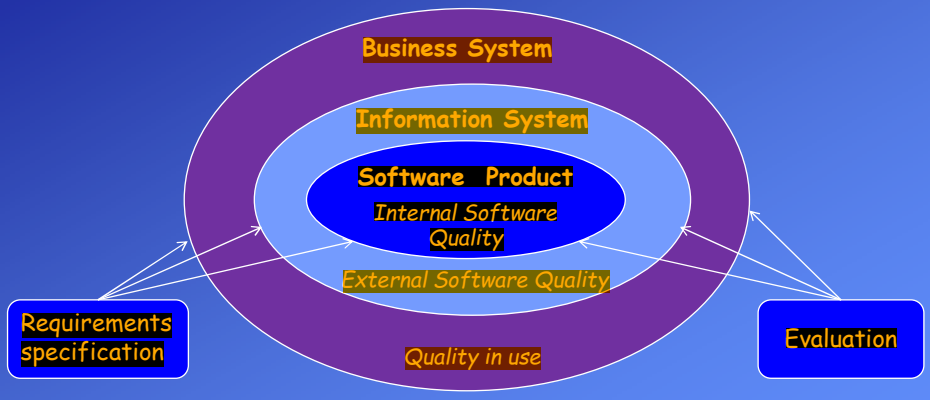
- Bir sistemin **kalitesi**, o sistemin ilgililerinin (*stakeholders*) (yatırımcı, proje lideri, tasarımcı, kullanıcı, müşteri, yazılımı kullanan firmanın yaptığı işten etkilenen kişiler vb.) gereksinimlerinin karşılanma miktarıdır.

  Buradaki gereksinimler hem belirtilmiş olanlar (*stated*) hem de doğal olarak olması beklenenlerdir (*implied*).
- Kalite standartlarının temel amaçları kalite modeli oluşturmada yol göstermek, ortak terimler oluşturmak ve unutulmaları önlemektir.
- Dokümanlarda yer alan kalite modelleri genel amaçlı olarak hazırlandıkları için modellerde yer alan unsurlar hepsi her proje için gerekli olmayabilirler.
- Bu nedenle kalite modelleri, üzerinde çalışılacak sisteme göre şekillendirilmelidirler.

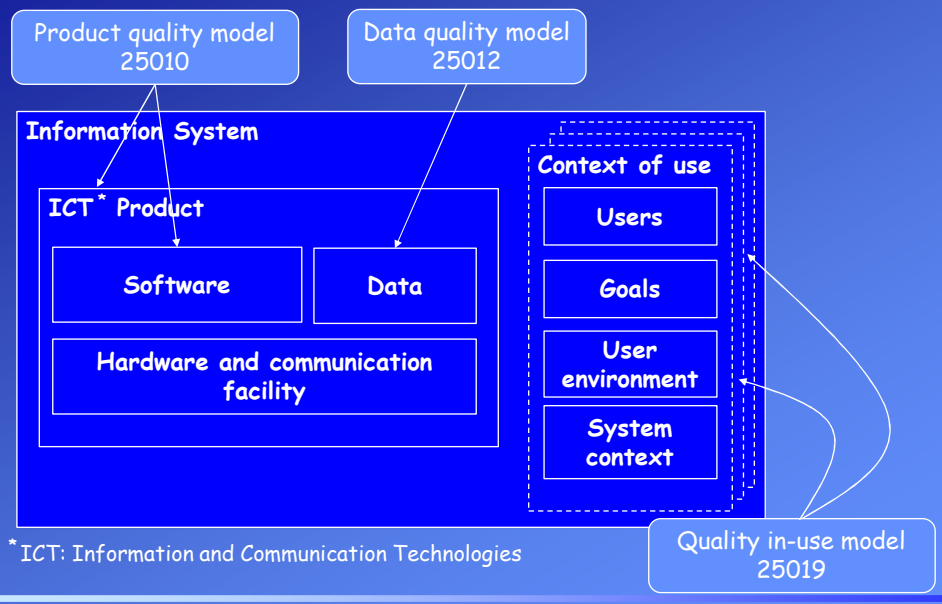
Amaç iki konuda yol göstermek:

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

Yazılım kalitesi farklı düzeylerde değerlendirilebilir:



Kalite modellerinin hedef unsurları (*target entity*)



*ICT: Information and Communication Technologies

Yazılımda ürün kalitesi yaşam döngüsünün modeli (*Software Product Quality Life Cycle Model*)

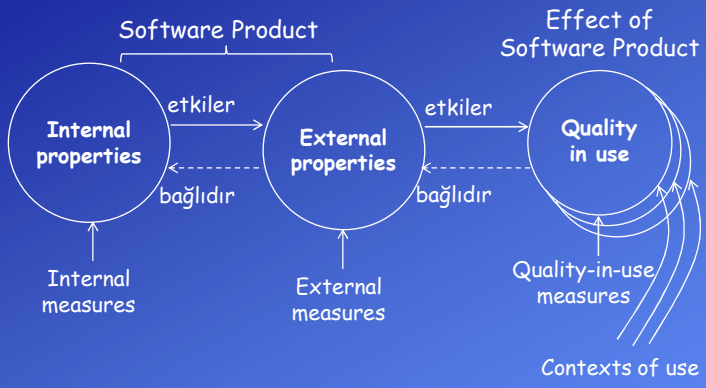
Yazılımın kalite yaşam döngüsü üç faza ayrılır:

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.

Yazılım sahada çalışırken kendisinden beklenenler (gereksinimler)
Son kullanıcı beklentileri ile ilgilidir.
Dış kalite gereksinimlerini belirlerler.

Program çalışırken teknik beklentiler:
1. İç kalite gereksinimlerini belirlerler.
2. Kullanımdaki kalite düzeyini kestirimde kullanılırlar.

Yazılımda kalite bileşenleri arasındaki etkileşim



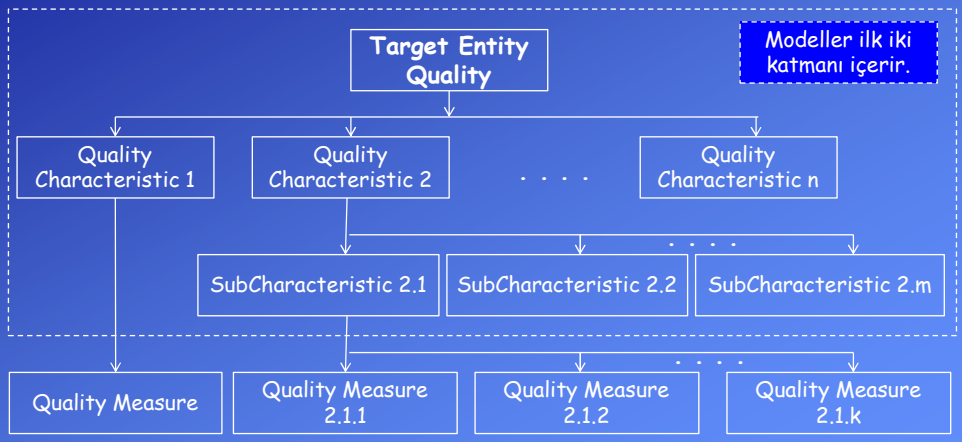
İç kalite nitelikleri (***Internal quality attributes***): Yazılımın iç yapısı (tasarım, kod) ile ilgili program çalışmadan (statik) ölçülebilen nitelikler.

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.

SQuaRE Kalite modelleri genel hiyerarşik yapısı

- Kaliteyi ana karakteristikler belirler.
- Bazı ana karakteristikler alt karakteristiklerden oluşur.
- Karakteristikler (varsa alt karakteristikler) kalite özelliklerinden (*quality property*) elde edilen ölçme değerleri (*quality measure*) ile değerlendirilirler.



ISO/IEC 25010 : 2023

Systems and software engineering —
Systems and software Quality Requirements and Evaluation (SQuaRE) —
Product quality model

Ürün kalite modeli:
Dokuz ana karakteristikten oluşur.

1. **Functional suitability**
   - Functional completeness
   - Functional correctness
   - Functional appropriateness
2. **Performance efficiency**
   - Time behavior
   - Resource utilization
   - Capacity
3. **Compatibility**
   - Co-existence
   - Interoperability
4. **Interaction capability**
   - Appropriateness recognizability
   - Learnability
   - Operability
   - User error protection
   - User engagement
   - Inclusivity
   - User assistance
   - Self-descriptiveness

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

5. **Reliability**
   - Faultlessness
   - Availability
   - Fault tolerance
   - Recoverability
6. **Security**
   - Confidentiality
   - Integrity
   - Non-repudiation
   - Accountability
   - Authenticity
   - Resistance
7. **Maintainability**
   - Modularity
   - Reusability
   - Analysability
   - Modifiability
   - Testability
8. **Flexibility**
   - Adaptability
   - Scalability
   - Installability
   - Replaceability
9. **Safety**
   - Operational constraint
   - Risk identification
   - Fail safe
   - Hazard warning
   - Safe integration

ISO/IEC 25019 : 2023

Systems and software engineering —
Systems and software Quality Requirements and Evaluation (SQuaRE) —
Quality-in-use model

Kullanımdaki kalite modeli (*quality-in-use model*):

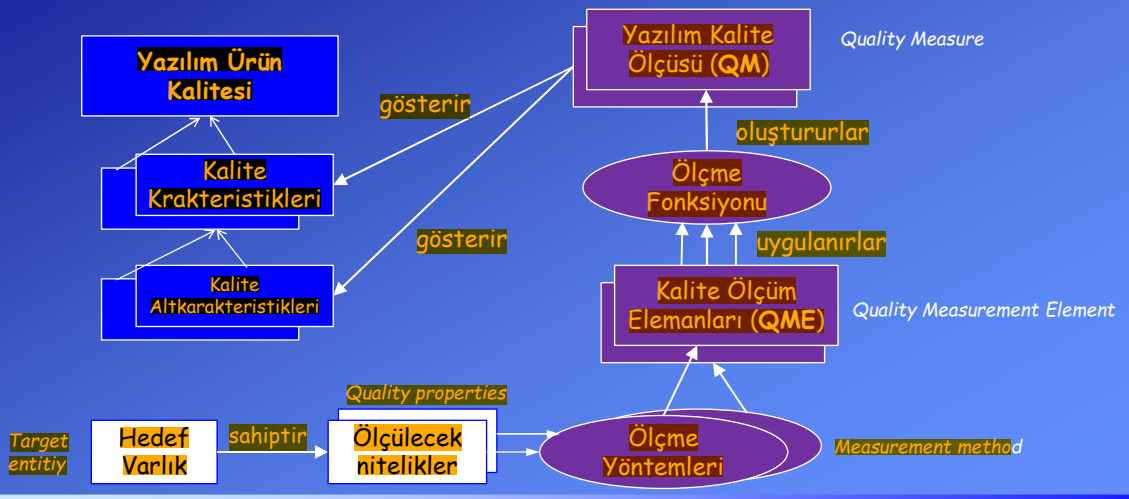
Karakteristiklerin anlamları ilgiliye (paydaş) (*stakeholder*) göre (birincil kullanıcı, bakım sorumlusu gibi) değişir.

1. **Beneficialness**
   - Usability
   - Accessibility
   - Suitability
2. **Freedom from risk**
   - Freedom from economic risk
   - Freedom from health risk
   - Freedom from environmental and societal risk
   - Freedom from human life risk
3. **Acceptability**
   - Experience
   - Trustworthiness
   - Compliance

ISO/IEC 25020 : 2019 Measurement reference model and guide

Software quality measurement reference model (SPQM_RM):

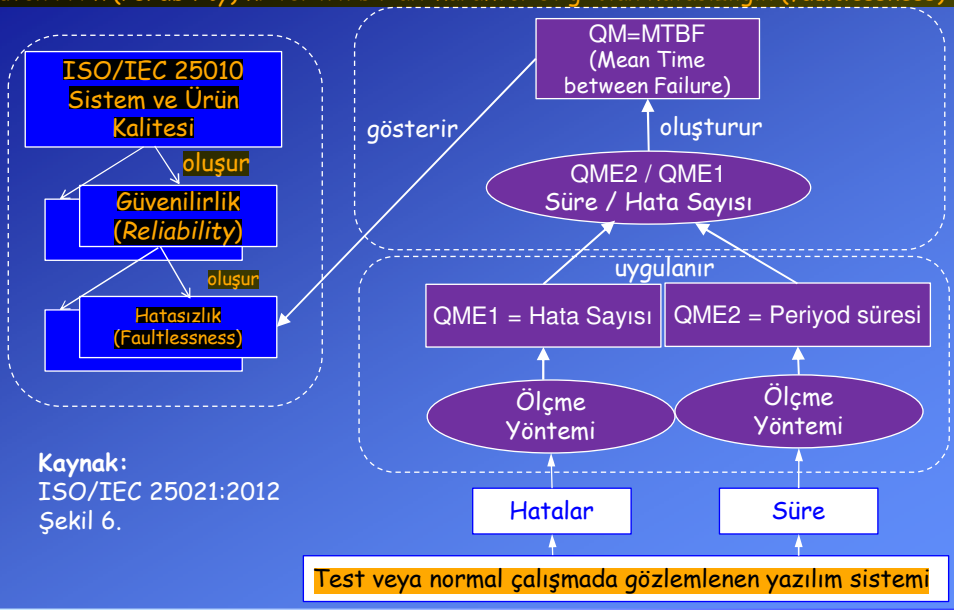
Kalite, karakteristikler , kalite özellikleri (nitelikleri) ve ölçme işlemleri arasındaki ilişkiyi gösterir.



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" (tek başına ifade edilebilen) 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, gözlem, uzmanlara sorma 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 hatasızlığın (*faultlessness*) ölçülmesi

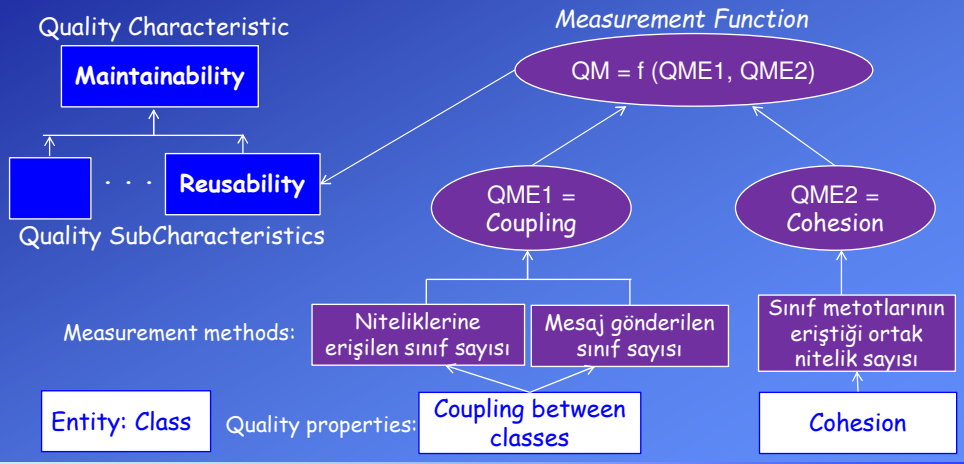


Kaynak:
ISO/IEC 25021:2012
Şekil 6.

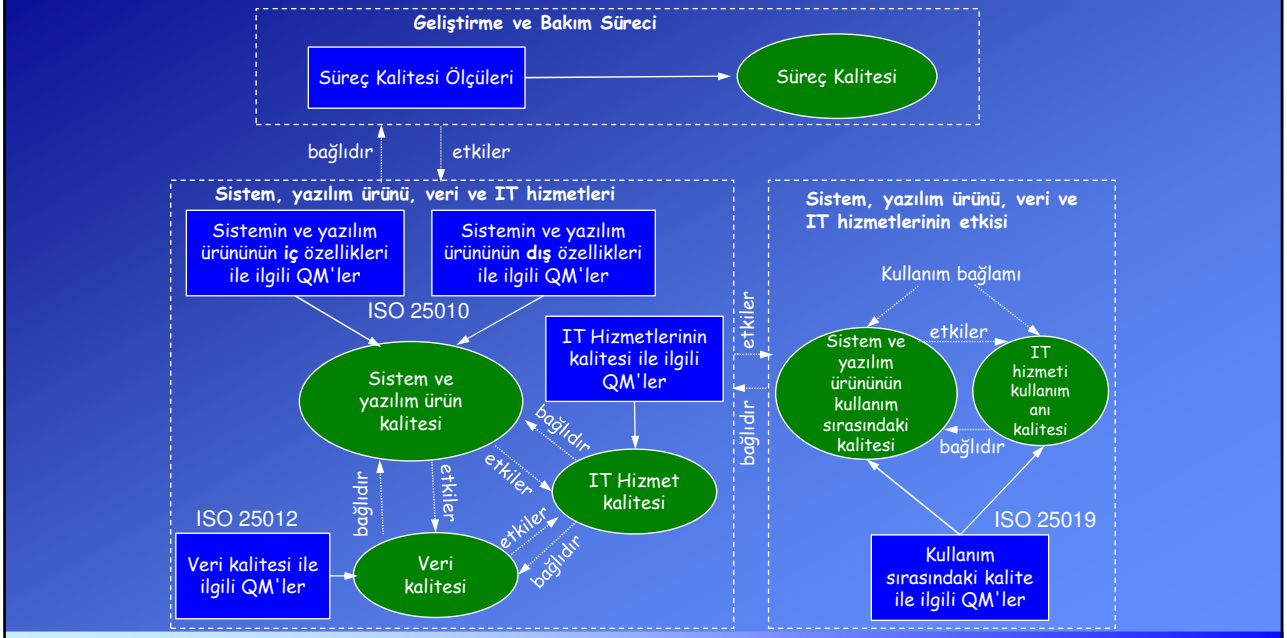
Örnek:

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*)