Yazılım Mühendisliğinde Ölçme

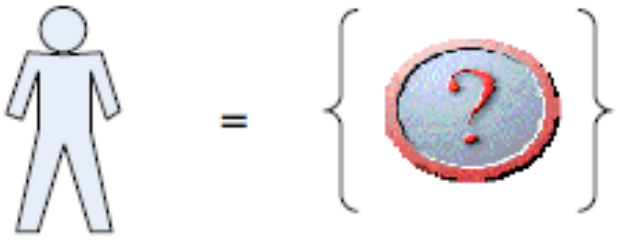
İçerik

- Yazılım mühendisliğinde ölçme
  - Ölçme teorisi ve ölçekler
  - Objektif ve sübjektif ölçevler
  - Temel ve türetilmiş ölçevler
- Yazılım ürün ölçevleri
- COSMIC İşlevsel Büyüklük Ölçme Yöntemi

Ölçme Nedir?

- Rakamları ve sembolleri <u>tanımlı kurallara göre</u> gerçek dünyadaki varlıkların özelliklerine atama işidir
  - Özellikleri <u>nicel olarak tanımlamak</u> amacıyla

$$\text{VARLIK} = \left\{ \begin{array}{l} \text{özellik}_1\ (\text{değer}_{11}, \text{değer}_{12}, \ldots) \\ \text{özellik}_2\ (\text{değer}_{21}, \text{değer}_{22}, \ldots) \\ \ldots \\ \text{özellik}_n(\text{değer}_{n1}, \text{değer}_{n2}, \ldots) \end{array} \right\}$$



Neden Ölçeriz?

Ölçmediğimiz şeyleri kontrol edemeyiz!

Diğer bir deyişle;

Kontrol etmek istediğimiz şeyleri ölçmek zorundayız!

Neleri Ölçeriz?

- Geçmişte ölçülemeyeceğini düşündüğümüz varlıkların özelliklerini bugün kolaylıkla ölçebiliyoruz
  - Zaman, sıcaklık, hız, zeka, enflasyon, vb.
- İlgi duyduğumuz ve anlamak istediğimiz şeylerin özelliklerini ölçme yollarını araştırırız

Ölçme Birimi

- Varlıkların özelliklerini sıklıkla, gerçek dünyayı algılayışımızı yansıtan rakam ya da semboller kullanarak tanımlarız
  - Örnek: santimetre olarak boy, beden olarak giysilerimiz

Ölçme ve Kestirme

Ölçme ("measurement"):

- Var olan bir varlığın bir özelliğine değer biçmek
  - Örnek: Yazılım kodlandıktan sonra kodun uzunluğunu (satır sayısını) bulmak

Kestirme ("estimation"):

- Henüz var olmayan bir varlığın bir özelliğini öngörmek
  - Örnek: Proje başlangıcında bir yazılım projesinin ne kadar süreceğini, maliyetini, vb. tahmin edebilmek

Objektif ve Sübjektif Ölçüler

- Objektif ölçever
  - Ölçme sonucunda farklı kişiler aynı değeri elde eder
    - Örnek: Metre ile bir duvarın yüksekliğini ölçmek
  - Nicel kontrolde daha etkinler

- Sübjektif ölçevler
  - Ölçme sonucunda elde edilen değer, ölçmenin yapıldığı ortama göre ve kişiden kişiye değişir
    - Örnek: Eğitim etkinliğini derecelendirmek
  - Kısıtlarını bildiğimiz sürece faydalanabiliriz

Temel ve Türetilmiş Ölçüler

- Bir varlığın özellik ölçevi, başka bir varlığı veya özelliği içermiyorsa (varlığın özelliğini direkt ölçüyorsa), temel ölçevdir
  - Örnek: Duvarın yüksekliği

- Bir ölçev, diğer ölçevler üzerinde yapılan işlemler sonucunda meydana geliyorsa, türetilmiş ölçevdir
  - Örnek: Sıvının yoğunluğu (kütle/hacim), arabanın hareket hızı (uzaklık/zaman)

Temel ölçevler ölçmenin yapı taşlarıdır.
Ancak pek çok yararlı ve etkin ölçev, türetilmiş ölçevler arasından çıkar!

Örnekler

- Temel ölçevler
  - Kod uzunluğu (“satır sayısı” ile ölçülür)
  - Test süresi (“saat” zaman birimi ile ölçülür)
  - Test hata sayısı (test boyunca tespit edilen hatalar sayılarak ölçülür)

- Türetilmiş ölçevler
  - Programcı üretkenliği (kod uzunluğu / kodlama işgücü)
  - Modül hata yoğunluğu (modül hata sayısı / modül kod uzunluğu)
  - Gereksinim değişiklikleri (değişen gereksinimlerin sayısı / ilk gereksinimlerin sayısı)

Ölçülebilecek Varlıklar

- Süreç
  - Belirli bir hedefe ulaşmak için gerçekleştirilen adımlar zinciri
    - Örnek: Yazılım gereksinim analizi
- Ürün
  - Bir süreç etkinliği ile üretilen çıktı
    - Örnek: Yazılım Gereksinimleri Tanımı Belgesi
- Kaynak
  - Bir süreç etkinliği tarafından kullanılan diğer bir varlık
    - Örnek: İnsan kaynağı

Yazılım Büyüklüğü (Ürün, İç Özellik)

4 boyutta ölçülebilir:

- **Uzunluk:** ürünün fiziksel büyüklüğü
  - Doküman:
    - Sayfa sayısı, karakter sayısı, tablo sayısı, şekil sayısı, vs.
  - Kod:
    - Satır sayısı (mantıksal veya fiziksel)
      - Sayma tekniği sebebiyle 5 kata kadar farklılık gösterebilir (Capers Jones, 1986)
    - Halstead uzunluk ölçümü
      - Uzunluk : N = N1 + N2 (N1: toplam operatör sayısı, N2: toplam değişken sayısı)
- **İşlevsellik:** yazılımın kullanıcıya sunduğu işlevselliğin miktarı
  - İşlev puan (“function points”), özellik puan (“feature points”), vs.
- **Karmaşıklık:** yazılımın çözdüğü konunun karakteristiği
  - Problem karmaşıklığı, algoritmik karmaşıklık (big-O notasyonu), yapısal karmaşıklık, kavramsal karmaşıklık
- **Tekrar kullanım:** yazılımın ne kadarı yeni geliştirildi?
  - Örnek: yüzde (%) olarak

Kod Uzunluğu

- Çoğu kez; yazılımın kalitesi, geliştirme işgücü ve maliyeti tahmini için temel girdi

- Sayma tekniğine göre kod satır sayısı (KSS), 5 kata kadar farklılık gösterebiliyor (Jones, 1986)
  - Örnek:

```
If A>B
    then A - B
    else A + B;
```

```
If A>B
    then
      begin
            A - B
      end
    else
      begin
            A + B
      end;
```

<u>Fiziksel sayma</u>:
Sol : 3 KSS
Sağ: 9 KSS

<u>Mantıksal sayma</u>:
Sol ve sağ: 1 KSS

<u>Anahtar sözcükleri sayma</u>
(if, then, else, begin, …):
Sol : 3 KSS
Sağ: 7 KSS

Kod Uzunluğu Ölçevleri

- Kaynak Kod Satır Sayısı (“Source Lines of Code”)
- Yorumsuz Kod Satır Sayısı (“Un-commented Lines of Code”)
- Yorum Kod Satır Sayısı (“Commented Lines of Code”)
- Yürütülür Deyim (“Executable Statements”)
- Program kaynak kodunun bilgisayarda tutacağı bayt miktarı
- Program kaynak kodunun içerisindeki karakter sayısı
- ....

Kod Uzunluğu

- Avantajlar
  - Otomatik olarak kolaylıkla sayılabilir
  - Direkt son ürünle ilgili

- Zorluklar
  - Belirsiz tanım
  - Dil bağımlı
  - Programcı bağımlı
  - Erken planlama için kullanılamıyor
  - Karmaşıklık gibi faktörleri adreslemiyor
    - Aynı uzunluktaki 2 kod parçasının genellikle aynı karmaşıklıkta olduğu varsayılıyor

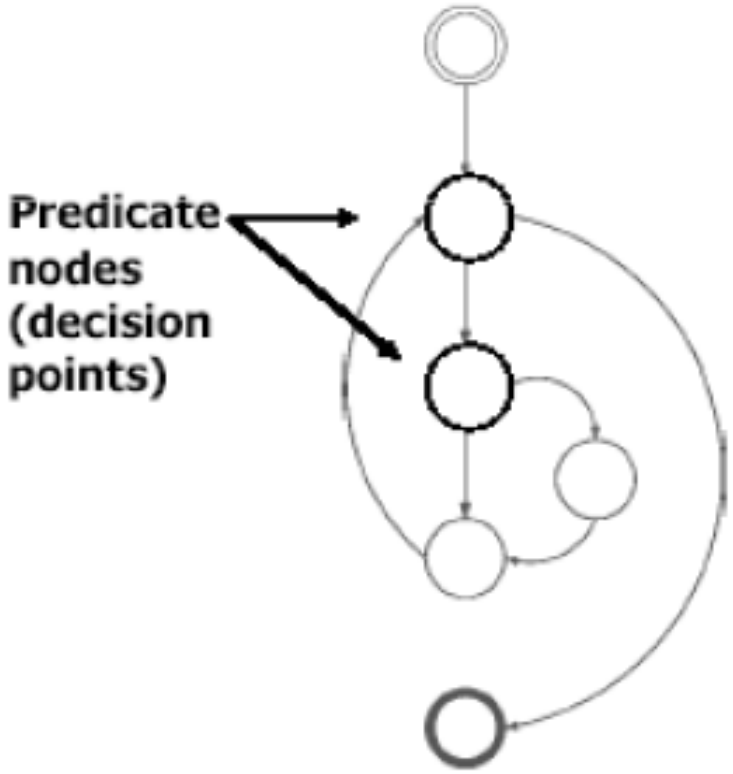
Yazılımın Yapısı (Ürün, İç Özellik)

3 tip altında incelenebilir:

- Kontrol-akış
  - Programdaki komutların işletiliş şekli
    - Örnek: sıra (“sequence”), seçim (“selection”), döngü (“iteration”)
- Veri-akış
  - Programla etkileşim halindeki verinin davranışı
- Veri-yapı
  - Programdan bağımsız olarak, veri yapıları
    - Örnek: “liste”, “yığın”, “kuyruk” kullanımı

Kontrol Akış Ölçevleri: Örnek McCabe's Cyclomatic Complexity

- Siklomatik karmaşıklık: v(F)
  - v(F) = e – n + 2
    - "n: node" (düğüm)
    - "e: edge" (geçiş)
  - Örnek:
    - $v(G) = e - n + 2$
    - $v(G) = 7 - 6 + 2$
    - $v(G) = 3$



Yazılımın Kalitesi (Ürün, Dış Özellik)

- Hata-esaslı ölçevler
  - Hata yoğunluğu (“defect density”) = Bilinen hatalar / ürün büyüklüğü
  - Sistem hasarı (“system spoilage”) = Müşteriye teslim sonrası bildirilen hataları düzeltmenin maliyeti / toplam proje maliyeti
- Bakım-yapılabilirlik ölçevleri
  - Ortalama tamir zamanı (“mean time to repair” - MTTR)
- Güvenilirlik ölçevleri
  - Ortalama bozulma zamanı (“mean time to failure” - MTTF)
  - Bozulmalar arası ortalama zaman (“mean time between failures” - MTBF)
    - MTBF = MTTF + MTTR
  - Mevcut olma zamanı (“availability” - A)
    - A = MTTF / (MTTF + MTTR) * 100

Yazılım İşlevsel Büyüklük Ölçme Yöntemleri

ISO Onaylı İBÖ Yöntemlerinin Ortak Özellikleri

- Yazılımın işlevsel büyüklüğünü "İşlevsel Kullanıcı Gereksinimleri"ni kullanarak hesaplarlar.

- İşlevsel Kullanıcı Gereksinimleri (İKG):
  - Kullanıcı gereksinimlerinin alt kümesidir
  - Kullanıcının ihtiyaç duyduğu ve yazılımın gerçekleştireceği kullanıcı etkinliklerini ve prosedürlerini ifade eder
  - Kalite ve teknik gereksinimleri dışarıda bırakır

- İşlevsel büyüklüğü ölçülecek yazılımın:
  - Geliştirilmesi / desteklenmesi için gereken yöntemlerden bağımsızdır
  - Fiziksel ve teknolojik bileşenlerinden bağımsızdır

International Function Point Users Group - (IFPUG) İşlev Puan Analiz Yöntemi

- Albrecht'in işlev puan analiz yöntemine dayanır [Albrecht, 1983].
- Genel tanımı verilen bir sistemin içerdiği işlevselliği tanımlar.
- İki adımda hesaplanır:
  - Ham işlev puan değerini hesaplama (HİP)
  - Teknik karmaşıklık faktörünü hesaplama (TKF)
  - Nihai işlev puan değerini hesaplama (İP)

    İP = HİP * TKF

Albrecht İşlev Puan Analiz Yöntemi

- Ham işlev puan değeri, aşağıdaki bileşenler cinsinden hesaplanır
  - Sistem girdilerinin sayısı (Ni)
    - Örnek: dosya isimleri, menü seçimleri, vs.
  - Sistem çıktılarının sayısı (No)
    - Örnek: Mesajlar, raporlar, vs.
  - Sistemin yanıtladığı sorgu sayısı (Nq)
    - Örnek: Hizmet fonksiyonları
  - Sistemin dış arayüz sayısı (Nef)
    - Örnek: Diğer sistemlerle etkileşim
  - Sistemin eriştiği iç varlık sayısı (Nif)
    - Örnek: Veritabanı, excel dosyası, vs.

Item	Weighting Factor		
	Simple	Average	Complex
External inputs (N_i)	3	4	6
External outputs (N_o)	4	5	7
External inquiries (N_q)	3	4	6
External interface files (N_{ef})	7	10	15
Internal files (N_{if})	5	7	10

- Örnek: Orta karmaşıklık → HFN = 4 Ni + 5 No + 4 Nq + 10 Nef + 7 Nif

Sayı	Bileşen Tipi	Ağırlık	Toplam
8	Girdi	8 x 4	32
12	Çıktı	12 x 5	60
4	Sorgu	4 x 4	16
2	Dış Arayüz	2 x 10	20
1	İç Varlık	1 x 7	7
	Ham Puan		135

Albrecht İşlev Puan Analiz Yöntemi

- Veri işleme uygulamaları için daha uygundur.
  - Gerçek zamanlı ve gömülü uygulamalar için Özellik Puan (“Feature Points”) yöntemi oluşturulmuştur.

- Kestirim yöntemi, Mark II ve COSMIC yöntemlerininkine göre daha kabadır.

Mark II İşlev Puan Analiz Yöntemi (1)

- Bir bilgi sistemi içindeki bilgi işleme yoğunluğunu, mantıksal işlembilgiler (Mİ) cinsinden ölçer [1998].
  - Metrics Practices Committee (MPC), UK Software Metrics Association
    - MK II Function Point Analysis Counting Practices Manual Version 1.3.1

- Bir Mİ, yazılım uygulaması tarafından desteklenecek en alt seviyedeki iş sürecidir. Üç bileşenden oluşur:
  - Uygulamaya veri sağlayan "girdi bileşeni"
  - Uygulama içindeki işlemleri yapan "işlem bileşeni"
  - Uygulama sonucunu sunan "çıktı bileşeni"

- Bir uygulamanın işlevsel büyüklüğü, bütün Mİ'lerinin işlevsel büyüklüklerinin toplamıdır.

- Bir Mİ kullanıcıyla ilişkili olan tek bir olay tarafından tetiklenir ve tamamlandığı zaman uygulamayı o olaya göre kendi içinde tutarlı bir durumda bırakır.

Mantıksal İşlembilgi

- "Müşteriye ait bilgilerin bakımını yap" bir Mİ midir? HAYIR
  - En alt düzeyde değildir, daha alt düzeydeki iş süreçlerinden oluşmuştur
    - Mİ: "Bir müşteriye ait bilgileri görüntüle"
    - Mİ: "Yeni bir müşteri ve o müşteriye ait bilgileri ekle"
    - Mİ: "Bir müşterinin görüntülenen bilgilerini sil"
    - Mİ: "Bütün müşterileri listele"

- "Bankamatikten para çekilmesi" bir Mİ midir? EVET
  - Tek bir olay tarafından tetikleniyor
  - Kullanıcıdan uygulama içine girdi var (müşteri kodu, para miktarı), içeride işlemler yapılıyor (çekilecek miktar hesaptan düşülüyor) ve kullanıcıya bir çıktı (makbuz) sunuluyor
  - Tamamladığı zaman uygulamayı tutarlı bir durumda bırakıyor

- Aşağıdakiler birer Mİ değil, çünkü iş süreci değiller
  - "Çekilecek para miktarının kullanıcı arayüzünden girilmesi"
  - "Hesaptan çekilecek para miktarının düşülmesi"
  - "Para sayma makinesine tetikleyici mesajın gönderilmesi"

Albrecht İşlev Puan Analiz Yöntemi

- Veri işleme uygulamaları için daha uygundur.
  - Gerçek zamanlı sistemler için kullanılması önerilmez.
- Sadece kullanıcı bakış açısını destekler.
  - Uygulamanın içindeki katmanları ve içteki İşlevselliği hesaba katmaz.
- Hesaplamadaki detay seviyesi sebebiyle, yazılım geliştirmenin erken aşamalarında (proje başlangıcında) kullanmak güçtür.
  - Ancak yazılım gereksinimleri analizi tamamlandıktan sonra bu detay yakalanabilir.

COSMIC İşlev Puan Analiz Yöntemi

- 1999 yılında Ortak Yazılım Ölçme Uluslararası Konsorsiyum'u (Common Software Measurement International Consortium-COSMIC) tarafından geliştirilmiştir.

- Yazılım uygulamasının fonksiyonel büyüklüğünü "Fonksiyonel Kullanıcı Gereksinimleri"ni temel alarak ölçmek üzere tasarlanmıştır.

- COSMIC İşlevsel Büyüklük Ölçme Süreci
  - Eşleme ("Mapping") Aşaması - Kurallar
  - Ölçme Aşaması - Kurallar

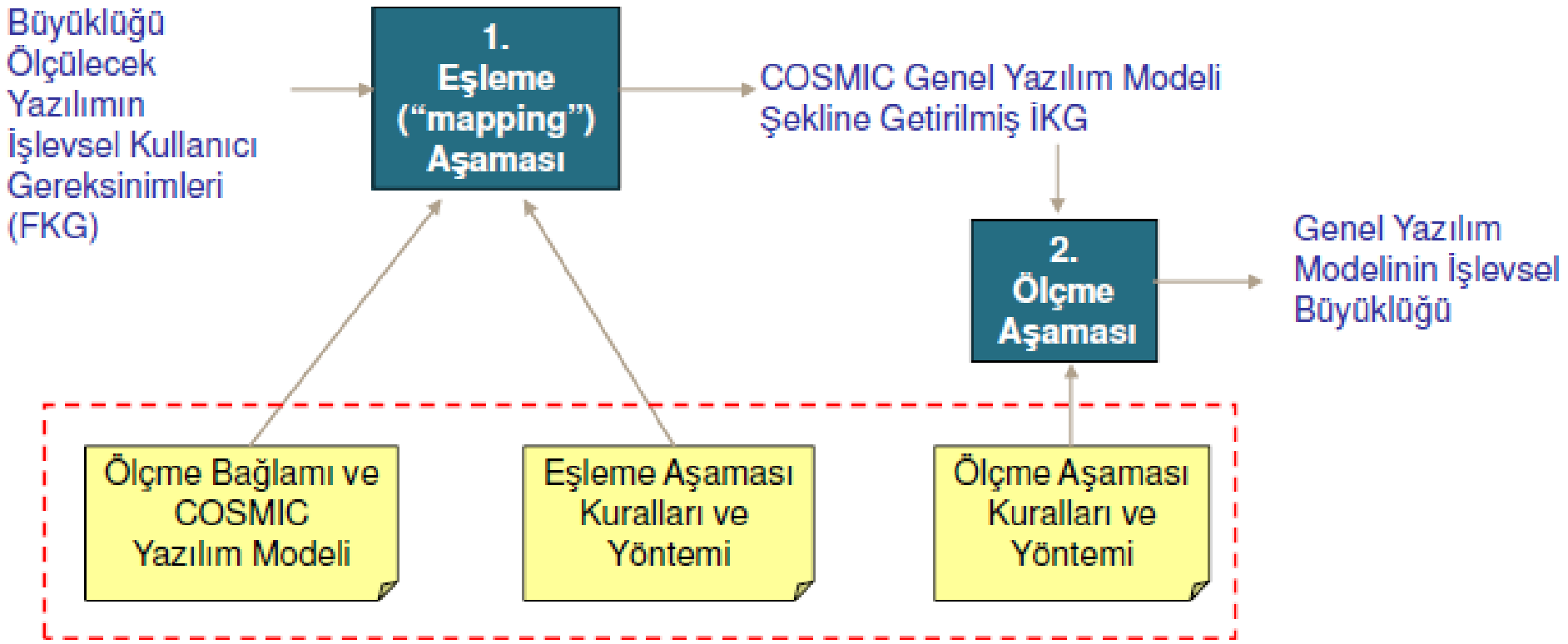
COSMIC İşlev Puan Analiz Yöntemi

- Uygulanabilirliği:
  - Veri-güçlü sistemler
    - Yönetim Bilgi Sistemleri, banka, muhasebe, personel, satın alma,vb.
  - Kontrol-güçlü sistemler (gerçek-zamanlı sistemler)
    - Avionik sistemler, çeşitli makineleri kontrol etmek üzere geliştirilen yazılım gömülü cihazlar (asansörler, çamaşır makineleri, vs.)
  - Melez (hibrid) sistemler
    - Gerçek zamanlı havayolu rezervasyon sistemleri, vs.
- COSMIC yöntemi ile ne ölçülebilir?
  - Yeni bir yazılım uygulamasının işlevsel kullanıcı gereksinimleri
  - Var olan bir yazılımda yapılacak değişikliklerin işlevsel kullanıcı gereksinimleri
  - Var olan ve şu anda kullanımda olan bir uygulamanın karşıladığı işlevsel kullanıcı gereksinimleri

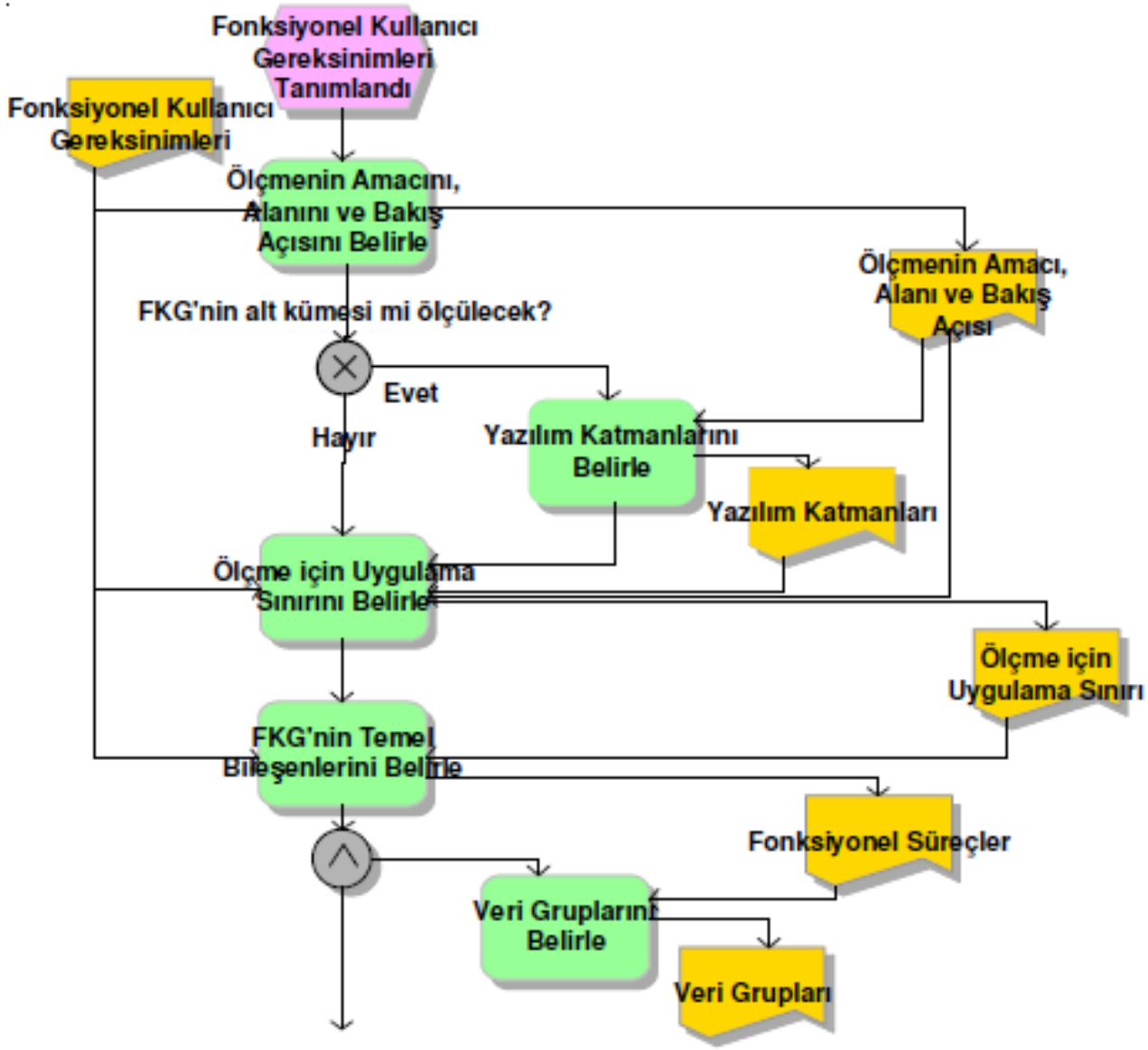
COSMIC İşlev Puan Analiz Yöntemi

- Yazılım yaşam döngüsünden bağımsızdır
  - Çağlayan (“waterfall”), Spiral, Artımlı (“incremental”), vb.
- Geliştirme metodolojisinden bağımsızdır.
  - Nesneye yönelik, vb.
- İşlevsel Kullanıcı Gereksinimleri (İKG)’nin bir ölçevidir, ancak bunların nasıl gerçekleştirildiğinden (“implementation”) bağımsızdır.

COSMIC Yöntemi Ölçme Süreci Modeli



1. Eşleşme Aşaması



“Ölçmenin Amacını, Alanını, Açısını Belirle”

- Ölçmenin amacı, ölçmenin neden yapıldığını ve sonucunun ne şekilde kullanılacağını anlatan bir cümledir. Örneğin;
  - Yazılım kestirim sürecine girdi olmak üzere,
  - İKG üzerinde anlaşmaya varıldıktan sonra, zaman içinde İKG ile ilgili yapılan değişikliklerin büyüklüğünün hesaplanması ve yönetilmesi,
  - Geliştiricilerin üretkenliğinin ölçülmesinde temel birim olarak,
  - ...
- Ölçme alanı, bir ölçme sürecinin kapsayacağı İKG kümesidir.
  - Bir yazılım uygulaması
  - Bir organizasyonun yazılım portföyü
  - ...
- Ölçmenin bakış açıları:
  - Son Kullanıcı Bakış Açısı: Yazılımın karşılayacağı ve/veya son kullanıcıya sağlayacağı işlevsellik miktarı.
  - Geliştirici Bakış Açısı: Yazılımın her bileşeninin karşılayacağı ve/veya sağlayacağı işlevsellik miktarı.

“Ölçme İçin Uygulama Sınırını Belirle”

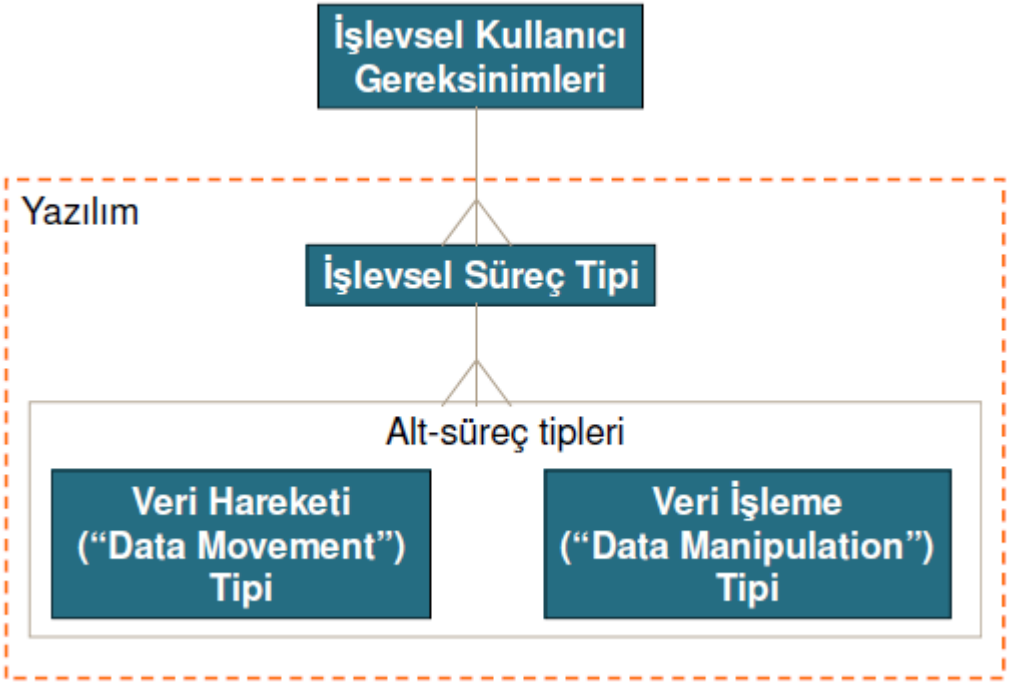
- Yazılım ile “kullanıcı” arasında kavramsal bir sınır tanımlar.
  - İş kullanıcısı – Bilgi girişi yapan ya da da çıktıları alan kişi (örneğin; müdür, terminal başındaki herhangi bir kişi)
  - Otomatikleştirilmiş kullanıcı – Bir başka uygulama yazılımı veya büyüklüğü ölçülen yazılım uygulamasından veri alan veya o yazılıma veri ileten mühendislik cihazları (örneğin; sensörler).
- Hangi işlevsel süreçlerin ölçüleceğinin ve diğer yazılım uygulamaları ile olan arabirimlerinin belirlenmesini sağlar.
- Ölçme sırasında hangi işlevselliklerin hesaba katılacağını hangilerinin ise dışarıda bırakılacağının belirlenmesini sağlar.

COSMIC Genel Yazılım Modeli (1)

- COSMIC genel yazılım modeli şu ilkelerin doğru olduğu varsayımına dayanır:

> 1. Eşlenecek ve ölçülecek yazılım, girdilerle beslenir ve kullanıcı için çıktılar üretir.
>
> 2. Eşlenecek ve ölçülecek yazılım, verileri çeşitli şekillerde işler.

- Bu modele göre, yazılım İşlevsel Kullanıcı Gereksinimleri, bir seri İşlevsel Süreç'e ayrıştırılır.
- Her İşlevsel Süreç veri hareketi ve/veya veri işleme (manipülasyon) icra eden tek ve eşsiz alt-süreçtir.



“İşlevsel Süreçleri Belirle”

- İşlevsel Süreç (“Functional Process”):
  - Bir set İKG’nin, kendi içinde bir bütün olan ve bağımsız olarak uygulanabilecek veri hareketlerinden oluşan temel bileşenidir.
  - Bir aktör tarafından dolaylı olarak veya bir ya da daha fazla olay tarafından doğrudan tetiklenebilir.
  - Tetikleyicinin ihtiyaç duyduğu bütün gereksinimi karşıladığında tamamlanmış olur.

- Tetikleyici olay: Uygulama sınırı dışında olan ve bir ya da daha fazla İşlevsel Süreci başlatan olaydır.

“İşlevsel Süreçleri Belirle”

- Kurallar:
  - Önce Tetikleyici Olayları belirler, sonra bu olaylar tarafından tetiklenen İşlevsel Süreçleri buluruz.
  - Tetikleyici Olay, ölçülen yazılımın “sınır”ı dışında oluşur ve bir ya da daha fazla İşlevsel Süreci başlatır. “Sınır”, Tetikleyici Olaylar ile İşlevsel Süreçler arasındadır.
  - Bir İşlevsel Süreci başlatan Tetikleyici Olay, daha alt seviyedeki olaylara bölünemez.
    - Saat ve zamanlamaya ilişkin olaylar, Tetikleyici Olay olabilirler.
    - Yazılım göz önünde bulundurulduğunda, bir Tetikleyici Olay ya olmuştur, ya da olmamıştır; yani anlıktır.

- Bir İşlevsel Süreç;
  - en azından bir adet İşlevsel Kullanıcı Gereksinimi’nden çıkarılmıştır
  - bir tetikleyici olay olduğu zaman icra edilir
  - en azından iki adet veri hareketi içerir;
    - 1 Giriş + 1 Çıkış / 1 Yazma

“İşlevsel Süreçleri Belirle”: Örnek-2

- “Bankamatikten para çekilmesi” bir İşlevsel Süreç midir?
  - EVET
    - Tek ve eşsiz bir olay tarafından tetikleniyor.
    - Uygulama sınırının içine doğru bir veri girişi var (para çekme isteği tetikleyicisi, müşteri kodu, para miktarı), içeride işlemler yapılıyor (çekilecek miktar hesaptan düşülüyor) ve kullanıcıya bir veri çıkışı (makbuz) var.
    - Tamamlandığı zaman Uygulamayı tutarlı bir durumda bırakıyor.
  - “Çekilecek para miktarının kullanıcı arayüzünden girilmesi” bir İşlevsel Süreç değildir.
  - “Hesaptan çekilecek para miktarının düşülmesi” bir İşlevsel Süreç değildir.
  - “Para sayma makinesine tetikleyici mesajın gönderilmesi” bir İşlevsel Süreç değildir.

“Veri Gruplarını Belirle”

- Veri Grubu: Aynı nesne ile ilişkili, farklı, boş olmayan ve sıralı olmayan veri özellikleri setidir.
  - Örnek: Müşteri (numarası, adı, soyadı, hesap numarası, vs.)

- COSMIC Yöntemi her İşlevsel Süreç’teki, her biri bir Nesne hakkındaki bir grup veri özelliğini (Veri Grubu) hareket ettiren Veri Hareketi’ni belirlemeye dayanır.

- Bunun için öncelikle Nesne’leri ve veri hareketlerini belirlemek için Varlık Analiz Yöntemlerinden birisini kullanmak gerekir.
  - Örnek: E/R diyagramları, Sınıf Diyagramları, vs.

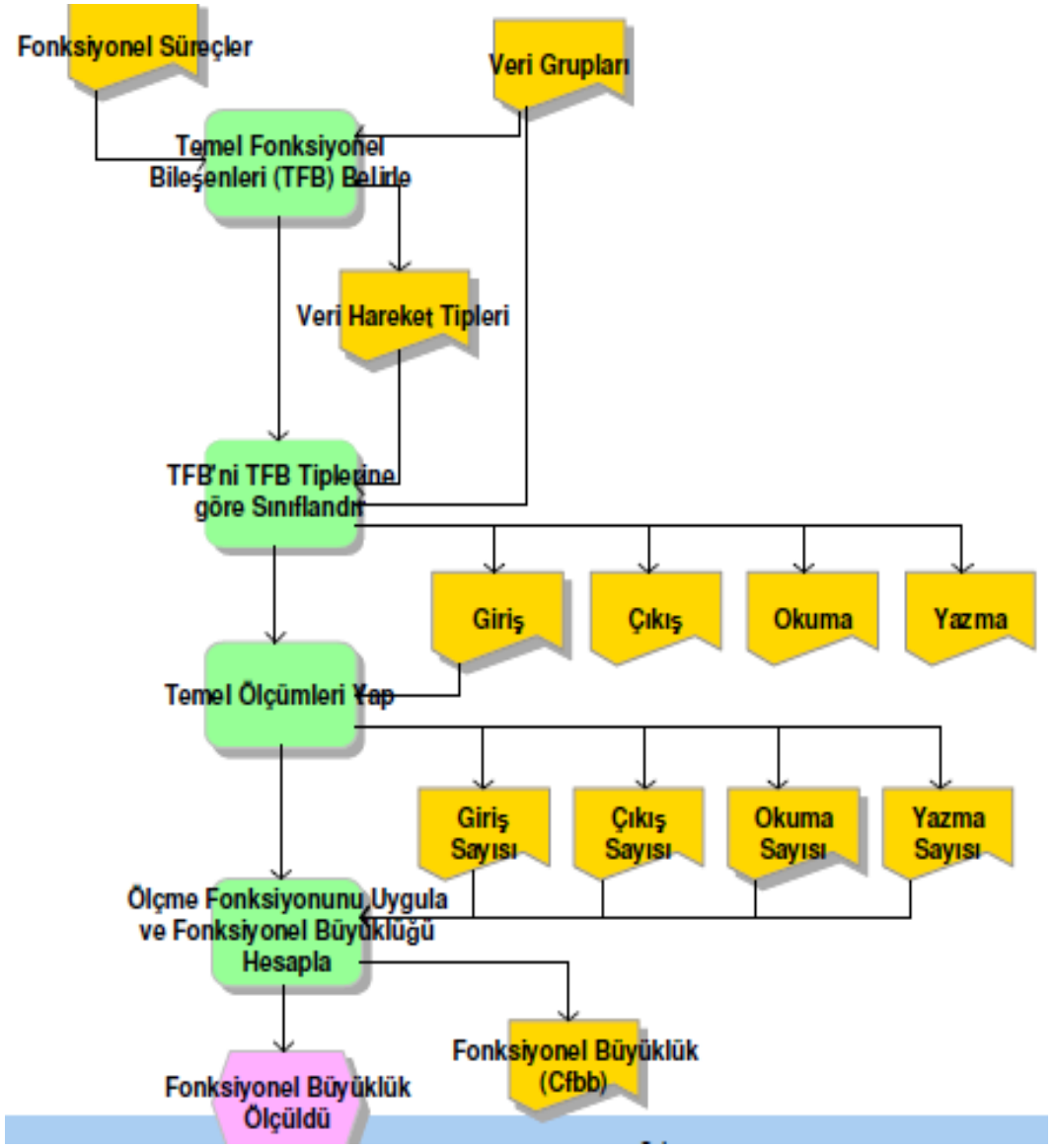
Veri Grubu - İlkeleri

1. Bir yazılımda Veri Grubu üç şekilde gerçekleşebilir:
   a. Sürekli depoda (persistent storage) (dosya, veritabanı tabloları,vb.);
   b. Değişken depoda (volatile storage)
   c. Girdi/Çıktı aygıtında (ekran, basılı rapor, vb.),

2. Belirlenen her Veri Grubu tek ve eşsiz olmalı ve içerdiği veri özellikleri ile ayırt edilmelidir.

3. Her Veri grubu, yazılımın İşlevsel Kullanıcı Gereksinimleri'nde tanımlı bir Nesne ile ilişkili olmalıdır.
   - Üç tipi vardır:
     - Geçici (Transient)
     - Sınırsız (Indefinite)
     - Kısa süreli (Short)

2. Ölçme Aşaması



Temel İşlevsel Bileşenleri Belirle ve Sınıflandır

- Temel İşlevsel Bileşen: Veri Hareketi (Data Movement):
  - Tek bir Veri Grubu Tipine dahil olan bir ya da daha fazla Veri Özelliği Tipini (Data Attribute Type) hareket ettiren Temel İşlevsel Bileşendir.
  - Bir İşlevsel Süreç yürütüldüğü sırada oluşur.

- Veri Hareketi Tipi dört çeşittir:
  - Giriş, Çıkış, Okuma,Yazma

Veri Hareketi Tipleri

- **Giriş:** Bir Veri Grubu'nu kullanıcıdan Uygulama Sınırı'ndan içeriye ihtiyaç duyulan İşlevsel Süreç'e doğru hareket ettiren Veri Hareketi Tipi'dir.

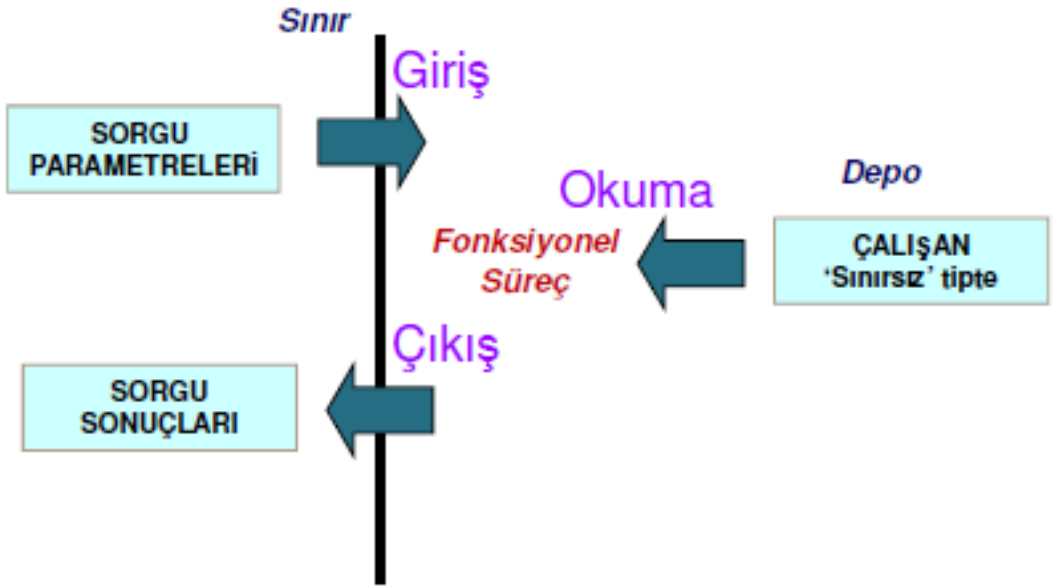
- **Çıkış:** Bir Veri Grubu'nu İşlevsel Süreç'ten Uygulama Sınırı'ndan dışarıya ihtiyaç duyan Kullanıcı'ya doğru hareket ettiren Veri Hareketi Tipi'dir.

- **Okuma:** Bir Veri Grubu'nu Sürekli Depo'dan (persistent storage) ihtiyaç duyulan İşlevsel Süreç'e doğru hareket ettiren Veri Hareketi Tipi'dir.

- **Yazma:** Bir Veri Grubu'nu içinde barındıran İşlevsel Süreç'ten Sürekli Depo'ya doğru hareket ettiren Veri Hareketi Tipi'dir.

Örnek

- İşlevsel Süreç: “Yazılım yaşı 35’den büyük olan çalışanların listesinin personel veritabanından okunarak kullanıcıya gösterilmesine olanak sağlayacaktır”



Temel Ölçümleri Yap

- Yazılım uygulamasının toplam işlevsel büyüklüğü, bütün İşlevsel Süreçlerin büyüklüklerin toplanmasıyla bulunur.
  - Bir İşlevsel Süreci oluşturan her veri hareketinin ne tipte olduğu belirlenir
    - (Giriş, Çıkış, Okuma, Yazma)
  - *İşlevsel Büyüklük Birimi*: COSMIC İşlevsel Büyüklük Birimi – Cfbb
  - Her veri hareketi tipinin büyüklüğü = 1 Cfbb

Ölçme Fonksiyonunu Uygula

- Bir İşlevsel Sürecin toplam işlevsel büyüklüğü, bütün veri hareketi tiplerinin işlevsel büyüklüklerinin toplanması ile bulunur.

$$\text{Büyüklük}_{Cfbb}\,(FS_i) = \Sigma\ \text{büyüklük}(\text{girişler}_i) + \Sigma\ \text{büyüklük}(\text{çıkışlar}_i) + \Sigma\ \text{büyüklük}(\text{okumalar}_i) + \Sigma\ \text{büyüklük}(\text{yazmalar}_i)$$