COSMIC İşlevsel Büyüklük Ölçme Yöntemi

Sürüm 3.0.1

Ölçme Kılavuzu

(ISO/IEC 19761: 2003 için COSMIC Uygulama Rehberi)

Ocak 2012

Teşekkürler

COSMIC Çekirdek Takım Yazarları, Sürüm 2.0[1] (alfabetik sıra ile)

Alain Abran, École de technologie supérieure – Université du Québec,
Jean-Marc Desharnais, Software Engineering Laboratory in Applied Metrics - SELAM,
Serge Oligny, Bell Canada,
Denis St-Pierre, DSA Consulting Inc.,
Charles Symons, Software Measurement Services Ltd.

Sürüm 2.0'yi Gözden Geçirenler 1998/1999 (alfabetik sıra ile)		
Moritsugu Araki, JECS Systems Research, Japan	Thomas Fetcke, Germany	Patrice Nolin, Hydro Québec, Canada
Fred Bootsma, Nortel, Canada	Eric Foltin, University of Magdeburg, Germany	Marie O'Neill, Software Management Methods, Ireland
Denis Bourdeau, Bell Canada, Canada	Anna Franco, CRSSM, Canada	Jolijn Onvlee, The Netherlands *
Pierre Bourque, , ÉCole de Technologie supérieure, Canada	Paul Goodman, Software Measurement Services,United Kingdom	Laura Primera, UQAM, Canada
Gunter Guerhen, Bürhen & Partner, Germany	Nihal Kececi, University of Maryland, United States	Paul Radford, Charismatek, Australia
Sylvain Clermont, Hydro Québec, Canada	Robyn Lawrie, Australia	Eberhard Rudolph, Germany
David Déry, CGI, Canada	Ghislain Lévesque, UQAM, Canada	Grant Rule, Software Measurement Services, United Kingdom*
Gilles Desoblins, France	Roberto Meli, Data Processing Organization, Italy	Richard Stutzke, Science Applications Int'l Corporation, United States
Martin D'Souza, Total Metrics, Australia	Pam Morris, Total Metrics, Australia*	Ilionar Sylva, UQAM, Canada
Reiner Dumke, University of Magdeburg, Germany	Risto Nevalainen, Software Technology Transfer Finland, Finland *	Vinh T. Ho, UQAM, Vietnam
Peter Fagg, United Kingdom	Jin Ng, Hmaster, Australia	

* COSMIC-FFP yazarları ile birlikte COSMIC Çekirdek Takımın kurucu üyeleri

[1] Sürüm 2.0 COSMIC-FFP Yönteminin ilk bilindiği şekli ile erişime açılmış birinci sürümüdür.

Sürüm 3.0'ü Gözden Geçirenler 2006/07 (alfabetik sıra ile)		
Alain Abran, École de Technologie Supérieure, Université du Québec, Canada	Jean-Marc Desharnais, Software Engineering Lab in Applied Metrics – SELAM, Canada	Arlan Lesterhuis*, Sogeti, The Netherlands
Bernard Londeix, Telmaco, United Kingdom	Roberto Meli, Data Processing Organization, Italy	Pam Morris, Total Metrics, Australia *
Serge Oligny, Bell Canada	Marie O'Neill, Software Management Methods, Ireland	Tony Rollo, Software Measurement Services, United Kingdom
Grant Rule, Software Measurement Services, United Kingdom	Luca Santillo, Agile Metrics, Italy	Charles Symons*, United Kingdom
Hannu Toivonen, Nokia Siemens Networks, Finland	Frank Vogelezang, Sogeti, The Netherlands	

* COSMIC Yöntemi Sürüm 3.0 ve 3.0.1'in editörleri

Sürüm 3.0.1'in Türkçe'ye Çevrilmesi 2008-2012 (alfabetik sıra ile)		
Selami Bağrıyanık, Turkcell Technology, Turkey	Pınar Efe, Siemens EC	Çiğdem Gencel, Free University of Bolzano, Italy
Barış Özkan, Middle East Technical University, Turkey	Mina Nabi, Middle East Technical University, Turkey	Ayça Tarhan, Hacettepe University, Turkey
Özden Özcan-Top, Middle East Technical University, Turkey	Oktay Türetken, Tilburg University, The Netherlands	Erdir Ungan, Middle East Technical University, Turkey
Gokcen Yılmaz, Middle East Technical University	Erhan Reşat Yüceer, STM Associates	

Türkçe çevirisine ilişkin herhangi bir yorumunuz veya sorunuz için lütfen Onur Demirors demirors@metu.edu.tr ile iletişime geçiniz.

COSMIC Ölçme Kılavuzu'nun ve diğer teknik raporların kamuya açık sürümlerine, diğer dillere çevirileri ile birlikte, www.gelog.etsmtl.ca/cosmic-ffp adresinden erişilebilir.

Sürüm Kontrolü

Aşağıdaki tablo bu dokümanın sürümlerinin tarihçesini vermektedir.

TARİH	GÖZDEN GEÇİREN(LER)	Değişiklikler / Eklemeler
99-03-31	Serge Oligny	Birinci taslak sürüm, gözden geçirenlerin yorumlarını almak üzere yayınlandı.
99-07-31	Teşekkür kısmına bakınız	Gözden geçirilip düzeltildi, gözden geçirenlerin yorumları ile birlikte.
99-10-06	Teşekkür kısmına bakınız	Gözden geçirilip düzeltildi, IWSM'99 çalıştayında[2] gözden geçirenlerin yorumları ile birlikte.
99-10-29	COSMIC Çekirdek Takımı	Gözden geçirilip düzeltildi, "alan deneme" sürüm 2.0 yayınlanmadan önceki son yorumlar.
01-05-01	COSMIC Çekirdek Takımı	ISO/IEC 14143-1: 1998'e uyum için + ölçme kurallarına açıklık getirmek üzere Sürüm 2.1'e güncellendi.
03-01-31	COSMIC Ölçme Pratikleri Komitesi	ISO/IEC FDIS 19761: 2002'ye uyum için + ölçme kurallarına ek açıklık getirmek üzere Sürüm 2.2'ye güncellendi.
07-09-01	COSMIC Ölçme Pratikleri Komitesi	Ölçme kurallarına ek açıklık getirmek ve özellikle Ölçme Stratejisi Evresi alanında diğer eklemeler için Sürüm 3.0'e güncellendi. Yöntemin adı "COSMIC-FFP yöntemi" yerine "COSMIC yöntemi" olarak değiştirildi. s2.2'den s.3.0'e güncellemelerde, Ölçme Kılavuzu s2.2'nin bölümleri diğer dokümanlara ayrıldı - aşağıdaki Önsöz bölümüne ve Ek D'ye bakınız.
09-05-01	COSMIC Ölçme Pratikleri Komitesi	Küçük yazınsal iyileştirmelerin ve açıklamaların yapılması ve örneklerin daha açık şekilde ayırt edilebilmeleri için Sürüm 3.0, Sürüm 3.0.1 ile revize edildi. Bu sürüm ayrıca "Yöntem Değişikliği Bültenleri" 3, 4 ve 5'te önerilen değişiklikleri de içermektedir. Bu değişikliklerin detayları için Ek D'ye bakınız.

Aşağıdaki tablo bu dokümanın Türkçe sürümlerinin tarihçesini vermektedir.

TARİH	GÖZDEN GEÇİREN(LER)	Değişiklikler / Eklemeler
02-01-2012	Türkçe çeviri takımı, teşekkür kısmına bakınız	Gözden geçirilip düzeltildi, tüm çeviri takımının yorumları ile birlikte.

[2] Proceedings of the International Workshop on Software Measurement IWSM'99, Lac Supérieur, Québec, Canada, September 8-10 1999. Detaylı bilgi için http://www.gelog.etsmtl.ca/iwsm99/index2.html adresine bakınız.

Önsöz

COSMIC yönteminin amacı, işlevsel alanları genelde 'iş uygulamaları' (veya 'Yönetim Bilgi Sistemleri') yazılımları ve 'gerçek-zamanlı' yazılımlar olarak bilinen yazılımların işlevsel büyüklüklerini ölçmek için standartlaşmış bir yöntem sağlamaktır.

COSMIC yöntemi Aralık 2002'de ISO/IEC JTC1 SC7 tarafından, ISO/IEC 19761 "Software Engineering – COSMIC-FFP – A functional size measurement method" (bundan sonra "ISO/IEC 19761" olarak geçecektir) uluslararası standardı olarak kabul edilmiştir.

ISO/IEC 19761, yöntemin temel normatif tanımlarını ve kurallarını içermektedir. Ölçme Kılavuzu'nun amacı sadece bu kuralları ve tanımları sağlamak değil, aynı zamanda ölçenlerin yöntemi tam olarak anlamaları ve uygulamaları için açıklamalar ve örnekler vermektir. Bununla birlikte, yöntem hakkında daha fazla deneyim kazandıkça, kurallar ve örnekler eklemek, hatta bazı temel kavramların tanımlarını geliştirmek değerli bulunmuştur. Common Software Measurement International Consortium (COSMIC), bu eklemelerin ve geliştirmelerin 2007/08 revizyonunda ISO/IEC 19761'e dahil edilmek üzere ISO'ya sunulmasını öngörmüştür.

Bu "Ölçme Kılavuzu" yöntemin 3.0 numaralı sürümünü tanımlayan dört COSMIC dokümanından biridir. Diğer üçü ise şunlardır:

- "Dokümantasyon Genel Yapısı ve Terimler Sözlüğü" (Sözlük, bütün COSMIC dokümanlarında ortak olarak geçen terimleri tanımlar. Bu doküman ayrıca, örnek olay çalışmaları ve alana özel yönergeler gibi diğer destek dokümanları da tanımlar).
- "Yönteme Genel Bakış"
- "İleri ve İlgili Başlıklar" (Bu doküman, büyüklük ölçümlerinin karşılaştırılabilmesini sağlama konusunu daha detaylı ele alacak ve daha önce bu Ölçme Kılavuzu'nun 2.2 numaralı sürümünde bulunan, erken veya hızlı yaklaşık büyüklük ölçmeye ve ölçümlerin çevrilebilirliğine ilişkin bölümleri içerecektir.)

İşlevsel Büyüklük Ölçme (İBÖ)'ye yeni başlayan veya başka bir İBÖ yöntemine aşina olan okuyuculara, bu Ölçme Kılavuzu'nu okumadan önce "Yönteme Genel Bakış" dokümanını okumaları önerilir.

COSMIC yöntemi sürüm 3.0 için ana değişiklikler

COSMIC yönteminin sürüm 2.2'den 3.0'e yükseltilmesi, bu sürümde önceki sürüme göre önemli bir gelişme olduğunu göstermektedir. Ölçme Kılavuzu sürüm 2.2'de tanımlı COSMIC yöntemi üzerinde 3.0 sürümünü oluşturmak üzere yapılan ana değişiklikler aşağıda verilmiştir.

- COSMIC yönteminin dokümanlarını daha kolay anlaşılır ve kullanılır yapmak amacıyla yöntem, yukarıda listelendiği gibi dört ayrı dokümana ayrılmıştır.
- Ölçme Kılavuzu'nun en son sürümü olan 2.2'ye kadar yayınlanmış olan iki "Yöntem Güncelleme Bülteni"ndeki (YGB) öneriler de bu sürüme eklenmiştir. Bunlar YGB 1 "Bir Yazılım Katmanının Tanımı ve Özelliklerine İlişkin İyileştirme Önerileri" ve YGB 2 "Bir İlgi Nesnesinin Tanımı İçin İyileştirme Önerileri"dir.
- Ölçme sürecinin birinci evresi olarak ayrı bir 'Ölçme Stratejisi' evresi tanımlanmıştır. Strateji evresi ayrıca, farklı yazılım parçalarının ölçümlerinin karşılaştırılabilmesini sağlamak amacıyla, ölçülecek yazılımın İşlevsel Kullanıcı Gereksinimleri (İKG)'nin 'tanesellik seviyesi' kavramını da dikkate alarak rehberlik sağlayacak şeklide iyileştirilmiştir.
- Deneyimler göstermiştir ki Ölçme Kılavuzu sürüm 2.2'de tanımlanan 'son kullanıcı' ve 'geliştirici' Ölçme Bakış Açısı kavramları, daha yalın bir kavram olan 'işlevsel Kullanıcı' kavramı ile değiştirilebilir. Bu ikinci kavram, basitçe, yazılımın İşlevsel Kullanıcı Gereksinimlerinde yer alan veri sağlayıcı veya veri alıcı taraf olarak tanımlanabilir. Bir yazılım parçası üzerinde yapılan bütün

ölçümler, o parçaya ait İşlevsel Kullanıcı Gereksinimleri ile yazılımın işlevsel kullanıcılarına sağlandığı belirtilen işlevsellik üzerinden gerçekleştirilir.

- Yöntemin ölçü birimi 'COSMIC işlevsel büyüklük birimi' (kısaca 'Cibb') yerine 'COSMIC İşlev Puan' (kısaca 'CİP') olarak değiştirilmiştir. Bu değişiklik okuma ve telaffuz kolaylığı ve yöntemin diğer 'İşlev Puan' yöntemleri ile uyumu için gerçekleştirilmiştir. Ayrıca, ek bir sadeleştirme olarak, yöntemin adı "COSMIC-FFP" yerine "COSMIC" olarak değiştirilmiştir.
- Sözlük, okunabilirliği artırmak için güncellenmiş, iyileştirilmiş ve yeni oluşturulan "Dokümantasyon Genel Yapısı ve Terimler Sözlüğü" dokümanına taşınmıştır.
- Veri analizi kavramları ile ilgili bazı kısımlar çıkartılmıştır. Bu kısımlar, COSMIC yönteminin uygulanışından çok, belirli bir alana özgü olduğundan, "İş Uygulamaları İçin COSMIC Ölçme Rehberi" dokümanına dahil edilmiştir.
- Terminolojide tutarlılığı ve anlaşılırlığı arttırmak amacıyla anlatıma ilişkin pek çok iyileştirme ve ekleme yapılmıştır. Değişikliklerden biri, muhasebe alanında yaygın olarak kullanılan, "kurallar, tanımlar ve ilkelerin uygulanmasına yardımcı olmak için vardır" anlayışından yola çıkarak yöntemin 'ilkelerini' ve 'kurallarını' birbirinden net bir şekilde ayırmak olmuştur. Hem ilkeler hem de kurallar uygulanması zorunlu kavramlar olarak görülmelidir. Bundan dolayı, ana metinde, ilke ve kural tanımlarında yer alan birçok örnek metinden çıkarılmıştır.

Yapılan bütün bu değişiklikler Ek D'de özetlenmektedir.

Ölçme Kılavuzu sürüm 2.2'ye aşina olan okuyucular, sürüm 3.0'de en çok değişikliği yeni ayrılmış olan 'Ölçme Stratejisi' evresinde göreceklerdir.

Yapılan ana değişikliklerin mevcut büyüklük ölçümleri üzerindeki sonuçları

Uluslararası Standardın oluşturulması sırasında ve Ölçme Kılavuzu sürüm 2.1, 2.2. ve 3.0'ün oluşturulmasında yapılan değişikliklere ve eklemlere karşın, COSMIC yönteminin 1999 yılında ilk olarak yayınlanan Ölçme Kılavuzu'nun birinci sürümünde (taslak) tanımlanan orijinal ilkeleri değiştirilmeden bırakılmıştır.

Ölçme Kılavuzu'nun 3.0 ve 3.0.1 numaralı sürümlerinde tanımlı ilkelere ve kurallara göre ölçülen işlevsel büyüklükler, sadece yeni kuralların daha kesin bir şekilde tanımlı olması sebebiyle, önceki sürümlerle yapılan büyüklüklerden farklı olabilir. Önceki sürümlere kıyasla kurallar, kişisel yorumlara daha az fırsat tanımaktadır. Kişisel yorumların azalmasından dolayı, ölçenler tarafından yapılan ölçümler arasında daha az fark olacaktır. Ayrıca Ölçme Stratejisi alanında yapılan değişiklikler ile ölçme biriminin adında yapılan değişiklik, ölçme sonuçlarının raporlanmasına ilişkin kurallarda önceki sürümlere göre basit bazı farklara neden olmuştur.

Ölçme Kılavuzu sürüm 2.2'den sürüm 3.0'e geçişte yapılan ana değişikliklerin açıklaması

Öncelikle vurgulamak gerekir ki COSMIC ölçme birimi; 'Cibb' (şimdiki adı ile 'CİP'), 1999 yılında ilk olarak kamuya açılan sürüm olan COSMIC-FFP Ölçme Kılavuzu'ndan bu yana hiç değiştirilmemiştir. Bu COSMIC'in bir standart uzunluk birimi olan metreye denklenmesi gibidir. Bu birim, metre gibi standart bir ölçü biriminin COSMIC'teki karşılığıdır. Ancak, bir yazılım parçasının işlevsel büyüklüğü çok farklı şekillerde ölçülebilir. Bu nedenle, herhangi bir İşlevsel Büyüklük Ölçme yöntemi için, "hangi büyüklüğü ölçmeliyiz?" sorusuna cevap vermek bazen sorun olabilir.

Buradaki birinci sorun şudur: Herhangi bir yazılım parçası çeşitli kullanıcılara işlevsellik sunmaktadır. 'kullanıcı' terimi, "ISO/IEC 14143/1 (İBÖ İlkeleri)" standardı terminolojisinde "ölçülecek yazılım ile etkileşen her şey" olarak tanımlanmıştır. Bu nedenle, bir yazılım parçasının işlevsel büyüklüğünün, kullanıcı(lar) olarak kimin veya neyin tanımlandığına bağlı olduğu söylenebilir. Örnek olarak bir mobil telefonun uygulama yazılımını ele alalım. Kullanıcılar; düğmelere basan insanlar, yazılım ile etkileşen donanım araçları (örneğin, ekran, tuşlar, vb.), uygulamayı destekleyen işletim sistemi veya ölçülecek uygulama ile etkileşen yazılım parçaları olabilir. Bu dört kullanıcı tipinin her biri, farklı işlevselliğe ihtiyaç duyar (bunun için işlevsel büyüklük, kullanıcı olarak kimin veya neyin tanımlandığına göre değişir). Bu durumda, verilen bir büyüklük ölçümü için, kullanıcıların kimler ve neler olduğunu, yani hangi işlevselliğin ölçülmüş olduğunu, nasıl bilebiliriz?

Bu sorun, COSMIC Ölçme Pratikleri Komitesi'nin (ÖPK) Ölçme Kılavuzu sürüm 2.2'de, 'son kullanıcı' ve 'geliştirici' Ölçme Bakış Açısı kavramlarını ortaya atmasına neden olmuştur. Ancak, deneyimler gösterdi ki bu tanımlar, özellikle de 'Geliştirici Ölçme Bakış Açısı', tüm ölçme ihtiyaçlarını tanımlamayı sağlayacak kadar genel değildi. ÖPK, doğru ve en genel yaklaşımın; 'işlevsel kullanıcı' kavramının tanımlandığı ve işlevsel büyüklüğün işlevsel kullanıcı olarak neyin veya kimin tanımlandığına göre değiştiği bir yaklaşım olduğu sonucuna vardı. İşlevsel kullanıcıların belirlenmesi, ölçmenin amacına bağlıdır ve işlevsel kullanıcılar normalde ölçülecek yazılımın İşlevsel Kullanıcı Gereksinimlerinden tespit edilebilir. Bu sebeple artık, belirli 'Ölçme Bakış Açılarının' tanımlanması fikrine gerek kalmamıştır.

İkinci bir sorun ise bir proje ilerledikçe İşlevsel Kullanıcı Gereksinimleri evrildiği için, büyüklük ölçümünün nasıl yapıldığına bağlı olarak, ölçülmüş büyüklüğün de büyüyor gibi görünebilmesidir. Yeni bir yazılım parçasının İKG'sinin birinci sürümü, 'üst seviyede' tanımlanmış olabilir. Proje ilerledikçe ve gereksinimler daha detaylı incelendikçe, İKG daha detaylı olarak veya 'alt seviyede' tanımlanır ve büyüklükleri artıyor gibi görünür. Biz bu farklı detaylandırma seviyelerini 'tanesellik seviyesi' olarak ayırt ediyoruz.

Bu durumda çözümlenmesi gereken sorun ise şudur: İki ayrı ölçmenin aynı 'tanesellik seviyesinde' yapılmış olduğundan nasıl emin olabiliriz? Ölçme Kılavuzu sürüm 3.0 bu konu hakkında, özellikle de bir projenin yaşam döngüsünün erken aşamalarında büyüklüğü ölçülecek İKG'nin henüz evrildiği dönem için, çok önemli tavsiyeler vermektedir. Bu konu, büyüklükler performans ölçümlerinde kullanıldığı ve karşılaştırma çalışmalarına özel amaçlarla diğer kaynaklardan alınan büyüklüklerle karşılaştırıldığı zaman, kritik hale gelmektedir.

Bu yeni 'işlevsel kullanıcı' ve 'tanesellik seviyesi' kavramlarının ve Ölçme Stratejisi'nde tanıtılan, bunları belirlemeye ilişkin süreçlerin, sadece COSMIC yöntemine özel olmadığını vurgulamak gerekir. Bunlar bütün İşlevsel Büyüklük Ölçme yöntemlerine uygulanabilir. COSMIC yöntemi sağlam mühendislik ilkelerini temel aldığı ve uygulanabildiği alanlar "1. nesil İBÖ yöntemlerine" kıyasla daha geniş olduğu için, "hangi büyüklüğü ölçmeliyiz?" probleminin düzgün bir şekilde tanımlanması gerektiği fark edilmiş ve bir çözüm üretilmiştir.

COSMIC'i, performansla ilgili (örneğin kestirim, kıyaslama vb. için) ölçümler yapmak amacıyla kullanan ölçenlerin, 'işlevsel kullanıcıları' veya ölçme yapılacak 'tanesellik seviyesini' belirlemek için çok fazla zaman harcamalarına gerek yoktur, zira bunlar genellikle çok belirgindir. Ancak, yapılacak seçimlerin açıkça ortada olmadığı durumlar için, Ölçme Kılavuzu'nun yeni bölümü olan "Ölçme Stratejisi Evresi" ile "İleri ve İlgili Başlıklar" dokümanının büyüklük ölçümlerinin karşılaştırılabilirliğini sağlamaya yönelik bölümü, bu konularda dikkate alınması gereken faktörleri sunmaktadır.

COSMIC Ölçme Pratikleri Komitesi

Mayıs 2009

İçindekiler

1

GİRİŞ

1.1 COSMIC Yönteminin Uygulanabilirliği

1.1.1 Uygulanabilir alanlar

COSMIC işlevsel büyüklük ölçme yöntemi aşağıdaki alanlarda yazılımın işlevselliğinin belirlenebilmesi için tasarlanmıştır:

- Veri-zengin yazılımlar. Bankacılık, sigortacılık, muhasebe, satın alma, insan kaynakları, dağıtım ya da üretim gibi iş yönetiminin desteklenmesi için gereken iş uygulaması yazılımları. Bu tip yazılımlar gerçek dünyadaki olaylara ilişkin çok miktardaki verinin yönetimini gerektirdiği için genellikle veri-zengin yazılımlar olarak nitelendirilmektedir.
- Gerçek-zamanlı yazılımlar. Gerçek dünyada gerçekleşen olaylara ayak uydurmayı ya da onları kontrol etmeyi gerektiren gerçek-zamanlı yazılımlar. Telefon santrali ve mesaj anahtarlama yazılımları; ev aletleri, asansörler, araba motorları ve uçak gibi makineleri kontrol eden cihazlara gömülü yazılımlar; süreç kontrolü ve otomatik veri toplama için kullanılan yazılımlar ve bilgisayarların işletim sistemlerinin içerisinde bulunan yazılımlar bu tür yazılımlara örnek olarak verilebilir.
- Melez yazılımlar. Yukarıda yer alan yazılımların karışımı olan yazılımlardır. Havayolları ya da oteller için geliştirilen gerçek zamanlı rezervasyon sistemleri bu tür yazılımlara örnek olarak verilebilir.

1.1.2 Uygulanabilir olmayan alanlar

COSMIC ölçme yöntemi henüz matematik-yoğun yazılımların işlevselliğini göz önünde bulunduracak şekilde tasarlanmamıştır. Burada matematik-yoğun yazılımlardan kasıt; uzman sistemler, benzetim sistemleri, kendi kendine öğrenen sistemler, hava tahmin sistemleri gibi karmaşık matematiksel algoritmalar ile nitelenen veya özel ve karmaşık kurallar içeren yazılımlar ya da bilgisayar oyunları ve müzik aletleri gibi sürekli değişkenleri işleyen yazılımlardır.

Öte yandan, bu tarz işlevsellik sağlayan yazılımlar için COSMIC ölçme yöntemine yerel açılımlar tanımlamak mümkündür. Ölçme Kılavuzu hangi durumlarda yerel açılımların kullanılabileceğini örnekler vererek açıklamaktadır. Yerel açılımlar Ölçme Kılavuzu'nun ölçüm raporlama bölümünde yer alan kurallara göre raporlandırılmalıdır.

1.1.3 İşlevsel büyüklüğü etkileyen faktörlerle ilgili kısıtlar

Bunlara ek olarak, uygulanabildiği alan içerisinde, yazılımın işlevsel büyüklüğünü ölçmeye yarayan COSMIC yöntemi, işlevselliğin yazılımın 'büyüklüğüne' etki edecek bütün olası yönlerini ölçmeye çalışmamaktadır. Örneğin, ne karmaşıklığın (tanımlamış olsa bile) ne de her veri hareketindeki veri özniteliği sayısının yazılımın işlevsel büyüklüğü üzerindeki etkisi, bu ölçme yöntemi tarafından değerlendirilmektedir (daha fazla bilgi için Ölçme Kılavuzu'nun Eşleme Evresine bakabilirsiniz). Bölüm 1.1.2'de değinildiği üzere istendiği takdirde, işlevsel büyüklüğün bu gibi faktörlerini, COSMIC ölçme yöntemine yerel bir açılım yaparak hesaplamak mümkündür.

1.1.4 Çok küçük yazılımları ölçmeyle ilgili kısıtlar

Tüm işlevsel büyüklük ölçme yöntemleri, ilgili uygulama alanı için 'ortalama' kabul edilebilecek, yazılımın işlevselliğinin basitleştirilmiş bir modeli üzerindeki varsayımlara dayanır. Çok küçük yazılım

parçalarının ölçümü, karşılaştırması ya da kullanımı (örneğin kestirimi) sırasında dikkatli olmak gerekir. Özellikle yazılımda meydana gelen küçük değişimleri ölçerken 'ortalama' varsayımı bozulabilir. COSMIC yöntemindeki 'çok küçük' kavramı 'birkaç veri hareketi' anlamına gelir.

1.2 İşlevsel Kullanıcı Gereksinimleri

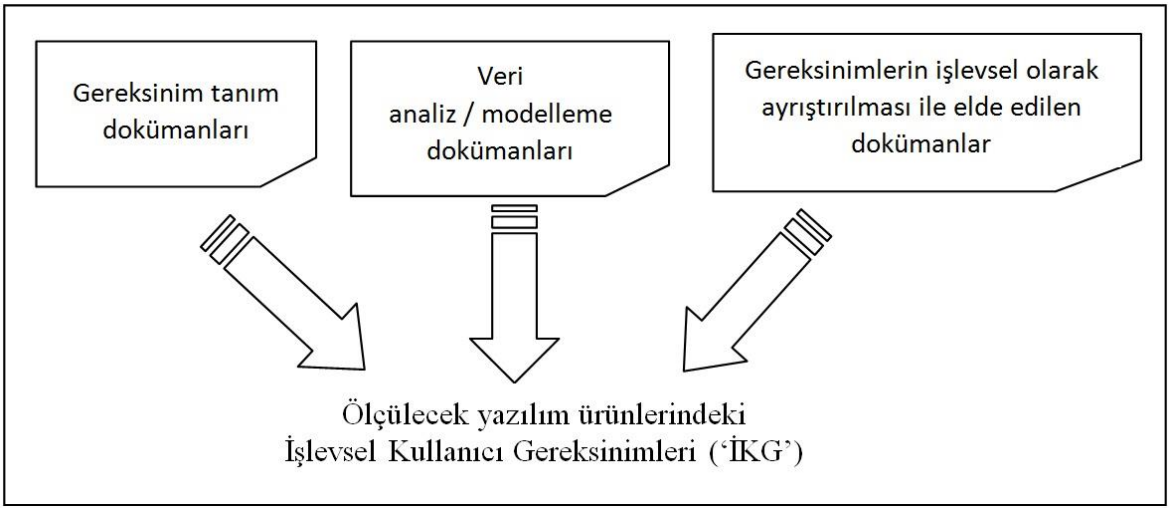
COSMIC ölçme yöntemi, ölçümü yapılacak yazılım parçasının **İşlevsel Kullanıcı Gereksinimlerine (İKG)** bir dizi modelin, ilkenin, kuralın ve sürecin uygulanmasını içerir. Elde edilen sonuç COSMIC yöntemine göre bir parça yazılımın işlevsel büyüklüğünün 'miktarının sayısal değerini' (ISO tarafından belirlendiği gibi) ifade eder.

COSMIC yöntemine göre ölçülen işlevsel büyüklükler, ölçümü yapılacak yazılımın operasyonel ürünlerindeki her türlü uygulama kararından bağımsız olacak şekilde tasarlanmıştır. İşlevsellik, 'yazılımın kullanıcıları için işleyeceği bilgi' ile ilgilidir.

Daha açık bir şekilde anlatmak gerekirse bir İşlevsel Kullanıcı Gereksinimi, bir yazılımın, yazılıma veri gönderen ya da yazılımdan veri alan işlevsel kullanıcıları için 'neler' yapması gerektiğini tanımlar. Bir İKG ifadesi, yazılımın 'nasıl' çalışması gerektiğini tanımlar; teknik ya da kalite gereksinimleri dışındaki gereksinimlerden oluşur (İKG'nin resmi tanımı için bölüm 2.2'ye bakınız). İşlevsel büyüklüğü belirlerken sadece İKG göz önünde bulundurulmalıdır.

1.2.1 Uygulamada yazılım ürünlerinden işlevsel kullanıcı gereksinimlerinin elde edilmesi

Gerçek yazılım geliştirme dünyasında, diğer yazılım gereksinimlerinden açıkça ayrıştırılmış ve herhangi bir yoruma gerek bırakmadan doğrudan ölçüme olanak sağlayacak biçimde tanımlanmış İKG çok nadir bulunur. Bu ise ölçenlerin genellikle İşlevsel Kullanıcı Gereksinimlerini, COSMIC 'yazılım modellerinin' kavramları ile ilişkilendirmeden önce, yazılım ürünlerinden çıkarması gerektiği anlamına gelir.



Şekil 1.2.1 – Gerçekleştirme öncesinde İşlevsel Kullanıcı Gereksinimleri

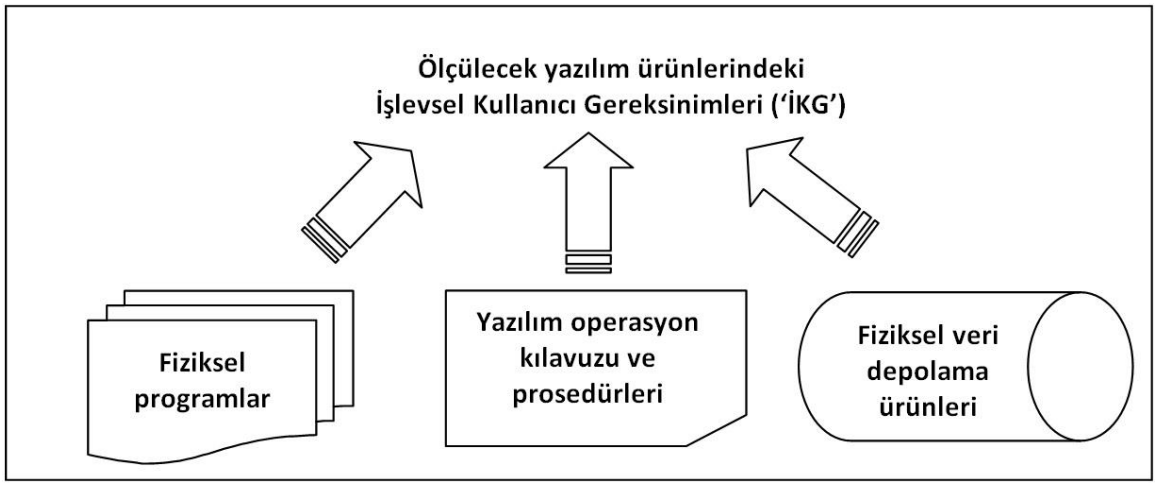
Şekil 1.2.1 de gösterildiği gibi İşlevsel Kullanıcı Gereksinimleri, yazılım geliştirilmeden önce yazılım mühendisliği ürünlerinden elde edilebilir. Bu nedenle yazılımın işlevsel büyüklüğü, yazılım geliştirilmeden önce de ölçülebilir.

Burada, yazılım ortaya çıkmadan önce oluşturulan ürünlerin (örneğin gereksinim toplama ya da analiz evresinde oluşanların), gereksinimler gelişmekte olduğu için yazılımı farklı detay seviyelerinde tanımlayabileceği belirtilmelidir (daha fazla bilgi için bölüm 2.4 e bakınız).

Not: İKG, donanımla ya da yazılımla ilişkilendirilmeden önce üretilebilir. COSMIC yöntemi yazılımın İşlevsel Kullanıcı Gereksinimlerini ölçmeyi amaçladığı için, sadece yazılımla ilişkilendirilen İKG ölçülür. Fakat ilkede COSMIC, nihai atama kararından bağımsız olarak, İKG yazılıma ya da donanıma atanmadan önce uygulanabilir. Örneğin, COSMIC yöntemini kullanarak bir cep hesap makinesinin işlevselliğinin büyüklüğünü, ürünün donanımı ya da yazılımı hakkında bilgi sahibi olmadan ölçmek mümkündür. Fakat COSMIC yönteminin yeni kurallar eklemeye gerek kalmadan, donanıma ait İKG'nin büyüklüğünü de ölçmede kullanılabileceğini güvence etmek için pratikte daha fazla test yürütmek gereklidir.

1.2.2 Kurulu yazılımdan işlevsel kullanıcı gereksinimlerini elde etmek

Bazı durumlarda, mevcut bazı yazılım parçalarının büyüklüğünü, herhangi bir mimari ya da tasarım ürününün bulunmadığı ya da İKG'nin belgelendirilmediği durumlarda ölçmek gerekebilir (örneğin miras yazılımlar). Bu tür durumlarda İKG'yi, yazılım geliştirildikten sonra Şekil 1.2.2'de gösterildiği gibi, kurulu yazılım ürünlerinden elde etmek mümkündür.



Şekil 1.2.2 – Gerçekleştirme sonrasında İşlevsel Kullanıcı Gereksinimleri

1.2.3 Yazılım ürünlerinden işlevsel kullanıcı gereksinimlerini elde etmek

Farklı yazılım mühendisliği ürünlerinden İKG'yi bulup çıkartmak ya da kurulu yazılımdan İKG'yi elde etmek ve bunları COSMIC yöntemi ile ölçebilmek amacıyla gerekli biçime getirebilmek için uygulanacak süreçlerin ve buna bağlı olarak gerekecek işgücünün farklılık göstereceği açıktır. İKG'yi elde etmek için uygulanacak süreçler, Ölçme Kılavuzu'nun kapsamı dışında kalmaktadır. Ölçümü yapılacak olan yazılımın İşlevsel Kullanıcı Gereksinimlerinin var olduğu ya da ölçümün amacı doğrultusunda yazılım ürünlerinden elde edilebileceği varsayılmaktadır.

Bu nedenle Ölçme Kılavuzu, COSMIC yazılım modelinin kavramlarını tanımlayacak ve anlatacak şekilde sınırlandırılmıştır (örneğin bulup çıkartma, elde etme ve türetmenin amaçları[3]). Bu kavramlar COSMIC yazılım modellerine ait kuralların içerisinde yer almaktadır – "Yazılım Bağlam Modeli" ve "Genel Yazılım Modeli".

Bu iki model, "Yönteme Genel Bakış" belgesinde anlatılmıştır. Modeller hakkında genel bir anlayışa ya da doğrulamaya ihtiyaç duyan okuyucular bu belgeye bakabilirler. Uyum sağlamak amacıyla modeller aşağıya kopyalanmış ve Ölçme Kılavuzu'nun 2. ve 3. bölümlerinde detaylı olarak anlatılmıştır.

[3] İş Uygulamaları İçin COSMIC Ölçme Rehberi, iş uygulamaları alanında kullanılan çok sayıda verinin ve gereksinim belirleme yönteminin COSMIC kavramlarına eşleştirilmesi konusunda yol göstermektedir.

1.3 COSMIC Yazılım Bağlam Modeli

COSMIC yöntemi ile ölçülecek olan yazılım (ölçme kapsamında) dikkatlice tanımlanmalı ve bu tanımda yazılımın diğer yazılımlar ya da donanımla olan etkileşimleri göz önünde bulundurulmalıdır. "Yazılım Bağlam Modeli", bu tanım için gerekli olan kavramları ve ilkeleri ortaya koymaktadır.

Bölüm 1.3 ve 1.4'te ilk kullanıldığında koyu renkle yazılmış olan terimler COSMIC yöntemine özgü olan anlamlarıyla verilmiştir. Resmi tanımlar için, "Dokümantasyon Genel Yapısı ve Terimler Sözlüğü" dokümanına bakabilirsiniz. Tüm bu tanımlar bölüm 2, 3, ve 4'te daha detaylı olarak anlatılmıştır.

İLKELER – COSMIC Yazılım Bağlam Modeli
a) Yazılım, donanım tarafından sınırlandırılmıştır.
b) Yazılım genel olarak **katmanlardan** oluşmaktadır.
c) Bir katman, bir ya da daha fazla birbirinden farklı **eş ("peer")** yazılım parçası içerebilir ve bu yazılım parçaları da ayrı eş bileşenlerden oluşabilir.
d) Ölçülecek yazılım parçası, ölçme **kapsamı**yla belirlenmeli ve kapsam tek bir katman ile sınırlı olmalıdır.
e) Ölçülecek yazılımın kapsamı, ölçmenin **amacına** bağlı olarak belirlenmelidir.
f) Yazılımdan veri alan ya da yazılıma veri gönderen **işlevsel kullanıcılar**, yazılımın işlevsel kullanıcı gereksinimlerinden yola çıkarak belirlenmelidir.
g) Yazılım, işlevsel kullanıcıları ile **sınırı** geçen **veri hareketleri** ve sınır içerisinde yer alan kalıcı bellek alanına veri gönderen ya da kalıcı bellek alanından veri alan yazılım parçası aracılığıyla etkileşimde bulunur.
h) Yazılımın İşlevsel Kullanıcı Gereksinimleri farklı **tanesellik seviyelerinde** verilmiş olabilir.
i) Ölçmenin yapılacağı tanesellik seviyesi, normal olarak **işlevsel süreçler** seviyesinde olmalıdır.
j) Eğer işlevsel süreçler bazında ölçme yapmak mümkün değilse, yazılımın İşlevsel Kullanıcı Gereksinimleri kestirim yaklaşımı ile ölçülmeli ve işlevsel süreçler tanesellik seviyesinde ölçeklenmelidir.[4]

Yazılım Bağlam Modelindeki kavramlar, Ölçme Kılavuzu'nun "2. Ölçme Stratejisi" bölümünde detaylandırılmıştır.

1.4 Genel Yazılım Modeli

Ölçümü yapılacak olan yazılımın İşlevsel Kullanıcı Gereksinimlerini "Yazılım Bağlam Modeli" kapsamında yorumladıktan sonra, ölçülecek işlevselliğin bileşenlerini belirlemek için İKG'ye "Genel Yazılım Modelini" uygulayabiliriz. "Genel Yazılım Modeli" aşağıdaki genel ilkelerin, yöntem ile ölçülecek her yazılım için doğru olduğunu varsaymaktadır. (Sözlükte belirtildiği gibi, herhangi bir İşlevsel Büyüklük Ölçme yöntemi veri ya da işlevlerin 'tiplerini' belirlemeyi amaçlar; 'ortaya çıkışlarını' değil. Bundan dolayı, aşağıdaki metinde 'tip' eki, COSMIC temelli kavramlardan bahsedilirken ve 'tip' ile 'ortaya çıkış' arasındaki farkı ayırmak gerekli olmadıkça göz ardı edilecektir.)

İLKELER – COSMIC Genel Yazılım Modeli
a) Yazılım, işlevsel kullanıcılarından **girdi** verisini alır ve işlevsel kullanıcıları için **çıktı** verisini ve/veya başka bir sonucu üretir.
b) Ölçülecek yazılımın işlevsel kullanıcı gereksinimleri birbirinden ayrı ve eşsiz işlevsel süreçlerle eşleştirilebilir.
c) Her işlevsel süreç **alt-süreçlerden** oluşmaktadır.
d) Her bir alt-süreç ya bir **veri hareketi** ya da bir **veri işlemedir**.
e) Her işlevsel süreç bir işlevsel kullanıcının belirlediği bir **olay** sonrasında, işlevsel kullanıcı tarafından bir **Giriş** veri hareketi ile tetiklenir.
f) Bir veri hareketi, tek bir **veri grubunu** hareket ettirir.
g) Bir veri grubu tek bir ilgi nesnesini tarif eden, eşsiz **veri öznitelik** setinden oluşur.
h) Dört tip veri hareketi vardır. Giriş, bir veri grubunu işlevsel kullanıcıdan yazılıma doğru hareket ettirir. **Çıkış**, bir veri grubunu yazılımdan işlevsel kullanıcıya doğru hareket ettirir. **Yazma**, bir veri grubunu yazılımdan kalıcı belleğe doğru hareket ettirir. **Okuma**, bir veri grubunu kalıcı bellekten yazılıma doğru hareket ettirir.
i) Bir işlevsel süreç en az bir Giriş veri hareketi ve bir Yazma ya da bir Çıkış veri hareketi içermelidir. Bu nedenle bir işlevsel süreç en az iki veri hareketinden oluşur.
j) Ölçme sürecinin bir varsayımı olarak, veri işleme alt-süreçleri ayrı olarak sayılmaz; veri işleme alt-süreçlerinin işlevselliklerinin, ilişkili oldukları veri hareketi tarafından hesaba katıldığı varsayılmaktadır.[4]

Genel Yazılım Modelinin kavramları, Ölçme Kılavuzu'nun "3. Eşleme Evresi" bölümünde detaylı olarak ele alınmıştır.

Ölçülecek İKG Genel Yazılım Modeli ile eşleştirildiğinde İKG, Ölçme Evresi süreçlerini (bölüm 4) kullanarak ölçülebilir. Ölçme sonuçları, "Ölçüm Raporlama" bölümünde yer alan eğilimlere göre raporlandırılmalıdır (bölüm 5).

[4] 'Yaklaşık ölçme' (örneğin doğru büyüklüğü ölçmek için gerekli olan detay bulunamadığında) ve farklı seviyelerdeki ayrıştırma konuları " İleri ve İlgili Başlıklar" isimli COSMIC yönteminde yer almaktadır.

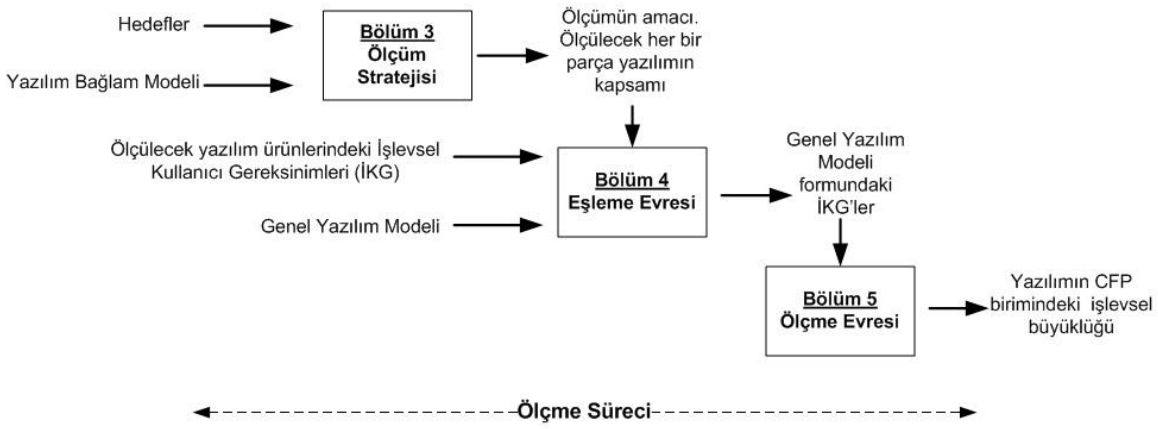
1.5 COSMIC Ölçme Süreci

Genel COSMIC ölçme süreci üç evreden oluşmaktadır:

- Ölçme Stratejisi, Yazılım Bağlam Modelinin ölçülecek yazılıma uygulandığı evredir (bölüm 2).
- Eşleme Evresi, Genel Yazılım Modelinin ölçülecek yazılıma uygulandığı evredir (bölüm 3).
- Ölçme Evresi, gerçek büyüklük ölçüm sonuçlarının elde edildiği evredir (bölüm 4).

Bir yazılım parçasına ölçme sürecinin uygulanması sonucunda, yazılımın İşlevsel Kullanıcı Gereksinimlerinin 'COSMIC İşlev Puan (CİP)' ile ifade edilmiş olan işlevsel büyüklüğü elde edilir.

COSMIC yönteminin bu üç evresi arasındaki ilişkiler, Şekil 1.5'te gösterilmiştir.



Şekil 1.5 – COSMIC yönteminin yapısı

2

ÖLÇME STRATEJİSİ EVRESİ

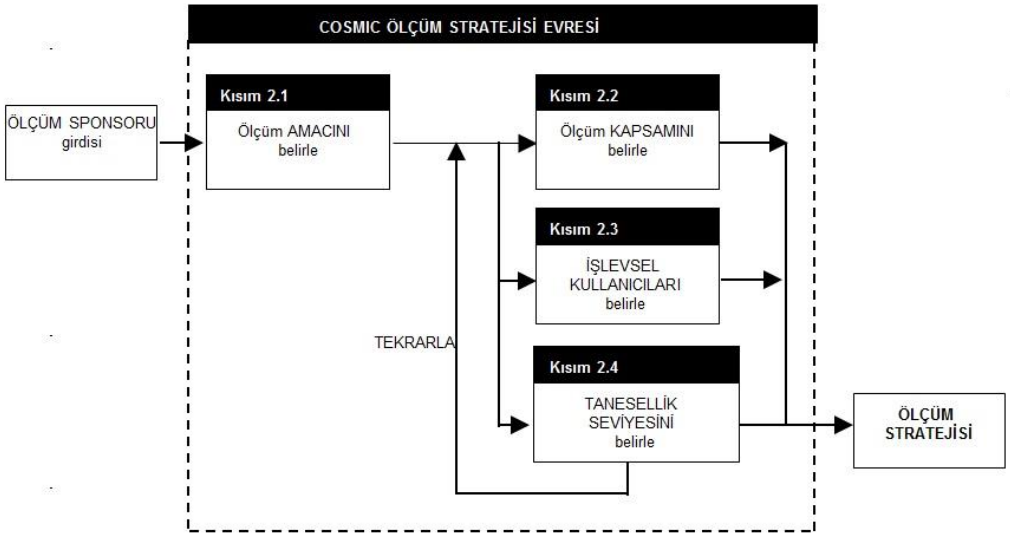
Bu bölüm, ölçmeye geçmeden önce düşünülmesi gereken ve yazılım işlevsel büyüklük ölçmenin dört anahtar parametresi olan amacı, kapsamı, işlevsel kullanıcıların belirlenmesini ve ölçülecek tanesellik seviyesini konu alır. Bu parametrelerin kararlaştırılması, "hangi büyüklük ölçülmeli?" ya da mevcut bir ölçüm için "bu ölçümü nasıl anlamalıyız?" sorularını yanıtlarken yardımcı olur.

Bu parametreler ve ilgili kavramlar yalnız COSMIC İşlevsel Büyüklük Ölçme yöntemine özgü değil, tüm İBÖ yöntemleri için ortak düşünülmelidir. Diğer İBÖ yöntemleri farklı işlevsel kullanıcıları, tanesellik seviyesini, vb. dikkate almayabilirse de COSMIC yönteminin daha geniş alanda uygulanabilirliği ve esnekliği, bu parametrelerin diğer İBÖ yöntemlerine göre daha dikkatlice düşünülmesini gerektirir.

Herhangi bir ölçüm sonucunu kaydederken, Ölçme Stratejisi evresinden gelen verinin kaydedilmesi (kısım 5.2'de listelendiği gibi) oldukça önemlidir. Bu parametrelerin tanımlanması ve kaydedilmesindeki başarısızlık, güvenilir şekilde anlaşılmayan ve karşılaştırılamayan ya da proje işgücü kestirimi gibi süreçlere girdi olarak kullanılamayan ölçümlere, sürekli olarak sebebiyet verecektir.

Bu bölümdeki dört kısım ölçene, Ölçme Stratejisini kararlaştırma sürecinde (Şekil 2.0'de gösterildiği gibi) yardımcı olmak üzere, her bir anahtar parametre için resmi tanımlar, ilkeler, kurallar ve örnekler verir.

Her kısım, anahtar parametrenin önemini anlatmak amacıyla ek açıklama içerir. Bu açıklama, parametrenin diğer ölçme alanlarında neden doğal olarak mevcut olduğunu benzerlikler kullanarak gösterir ve buradan yola çıkarak bu parametrenin yazılım işlevsel büyüklük ölçümünde de dikkate alınması gerektiğini belirtir.



Şekil 2.0 - Ölçme Stratejisi belirleme süreci

2.1 Ölçme amacının tanımlanması

Amaç terimi, normal Türkçe anlamında kullanılmıştır.

TANIM – Ölçme amacı
Ölçmeye neden ihtiyaç duyulduğunu, sonucun ne için kullanılacağını tanımlayan ifade

2.1.1 Ölçme amacı – bir benzerlik

Yazılımın işlevsel büyüklüğünün ölçülmesi için, tıpkı bir evin yüzey alanlarının ölçülmesinde olduğu gibi, birçok neden vardır. Amaç yeni geliştirilecek bir yazılımın maliyetini kestirmekse, tıpkı bir evin yüzey alanlarını inşasından önce ölçmek gibi, yazılımın işlevsel büyüklüğünü yazılım geliştirilmeden önce ölçmek gerekli olabilir. Farklı bir bağlamda ise, tıpkı planlara uygunluğu kontrol etmek için evin yüzey alanlarını inşasından sonra ölçmek gibi, kestirilen ve gerçekleşen maliyetleri karşılaştırmak için yazılımın işlevsel büyüklüğünü üretime alındıktan sonra ölçmek gerekli olabilir. Ölçmenin yapılma sebebinin, ölçü birimini ya da ölçme ilkelerini etkilemeksizin neyin ölçülmesi gerektiği üzerinde, bazen anlaşılması zor da olsa bir etkisi vardır.

Yukarıdaki ev örneğinde inşasından önce yüzey alanlarının ölçülmesi, açık şekilde ev planlarına dayanmaktadır. Gerek duyulan boyutlar (en ve boy) planlardan, uygun ölçeklendirme anlaşmaları kullanılarak çıkarılır ve yüzey alanları, iyi tanımlanmış anlaşmalara uygun olarak hesaplanır.

Benzer şekilde, yazılımın işlevsel büyüklüğünün geliştirmeden önce ölçülmesi, yazılımın planlarından yani geliştirmeden önce oluşturulan ürünlerinden türetilen, yazılımın İşlevsel Kullanıcı Gereksinimlerine dayanır. İKG bu ürünlerden, uygun anlaşmalar kullanılarak türetilir ve büyüklüğün ölçülebilmesi için gerekli boyutlar (veri hareketi sayısı) belirlenir.

Ev inşası benzerliğine devam edecek olursak yüzey alanlarının inşasından sonra ölçülmesi, farklı bir ölçme süreci gerektirir. Artık, gerek duyulan boyutlar (en ve boy), yapının kendisinden farklı bir araçla (mezura ile) çıkarılır. Ancak, ölçülen nesneler farklı olsa da (planları yerine yapının kendisi) boyutlar, ölçü birimi (herhangi bir ölçekleme anlaşmasıyla beraber) ve ölçme ilkelerinin tamamı aynı kalır.

Benzer şekilde, işlevsel büyüklüğün yazılım üretime alındıktan sonra ölçülmesi, gerekli boyutlar yazılımın kendi ürünlerinden çıkarıldığında, farklı bir ölçme sürecini gerektirir. Bu ürünlerin doğası farklı olsa da boyutlar, ölçü birimi ve ilkeler aynı kalır.

Ölçülecek nesnenin, sahibinin sözsel olarak anlattığı ev mi, planlarda tanımlanan ev mi ya da inşa edilmiş olan ev mi olacağı kararı ve ölçme için en uygun ürünlerin seçimi, ölçmenin amacına bağlı olarak ölçene kalmıştır. Açıkça, üç büyüklük birbirinden farklı olabilir. Aynı mantık yazılımın büyüklüğünü ölçerken de geçerlidir.

ÖRNEKLER: Aşağıdakiler tipik ölçme amaçlarıdır:

- İKG'nin büyüklüğünü evrildikçe, geliştirme işgücünü kestirme sürecine girdi olmak üzere ölçmek
- Başlangıçta kararlaştırılan İKG'ndeki değişikleri, kapsamı yönetmek için ölçmek
- Teslim edilen yazılımın İKG'nin büyüklüğünü, yazılım geliştiricinin performans ölçümüne girdi olmak üzere ölçmek
- Teslim edilen yazılımın İKG'nin toplam büyüklüğü ile geliştirilen yazılımın İKG'nin büyüklüğünü, işlevsel yeniden kullanım için bir ölçü elde etmek üzere ölçmek
- Mevcut yazılımın İKG'nin büyüklüğünü, yazılımın bakım ve desteğinden sorumlu olanların performans ölçümüne girdi olmak üzere ölçmek
- Mevcut yazılımın (İKG) üzerinde yapılan değişiklikleri, geliştirme proje takımının iş çıktısı büyüklüğü için ölçü oluşturmak üzere ölçmek
- Mevcut yazılımın insan işlevsel kullanıcılarına sağladığı işlevselliği ölçmek

2.1.2 Amacın önemi

Amaç ölçene şunları kararlaştırmada yardımcı olur:

- Ölçülecek kapsam ve ölçme için gerekli ürünler
- İşlevsel kullanıcılar (bölüm 2.3'de gösterileceği gibi, işlevsel büyüklük neyin ve kimin işlevsel kullanıcı olarak tanımlandığına bağlı olarak değişir)
- Proje yaşam döngüsünde ölçmenin gerçekleşeceği zaman
- Ölçmenin gereken kesinliği ve ilgili olarak COSMIC ölçme yönteminin ya da türetilmiş yerel yaklaştırma versiyonunun kullanımı (örneğin, proje yaşam döngüsünün erken aşamalarında, İKG tamamen belirlenmeden). Burada bahsedilen her iki nokta, İKG'nin ölçüleceği tanesellik seviyesini belirleyecektir.

2.2 Ölçme kapsamının belirlenmesi

TANIM – Ölçme Kapsamı
Bir işlevsel büyüklük ölçmede içerilecek İşlevsel Kullanıcı Gereksinimleri kümesi NOT: Amaca uygun olarak ölçülecek tüm yazılımların 'genel kapsamı' ile bağımsız olarak ölçülecek her yazılım parçasının genel kapsam içinde yer alan ve ayrıca ölçülmesi beklenen 'kapsamı' ayrıştırılmalıdır. Bu ölçme elkitabında 'kapsam' terimi (ya da 'ölçme kapsamı' ifadesi), ayrıca ölçülecek bağımsız yazılım parçası ile ilgilidir.

İşlevsel Kullanıcı Gereksinimlerini ISO aşağıdaki gibi tanımlanmıştır:

TANIM – İşlevsel Kullanıcı Gereksinimleri (İKG)
Kullanıcı gereksinimlerinin alt kümesi. Yazılımın işler ve hizmetler cinsinden ne yapması gerektiğini tanımlayan gereksinimler. NOT: İşlevsel Kullanıcı Gereksinimleri, aşağıdakilerle kısıtlı olmamakla birlikte, şunları içerir: • veri transferi (örneğin müşteri verisi girme, kontrol sinyali gönderme) • veri dönüşümü (örneğin banka faizi hesaplama, ortalama sıcaklığı türetme) • veri saklama (örneğin müşteri siparişini saklama, ortam ısısını kaydetme) • veri erişimi (örneğin mevcut çalışanları listeleme, uçağın en son pozisyonunu alma) İşlevsel olmayan Kullanıcı Gereksinimleri, aşağıdakilerle kısıtlı olmamakla birlikte, şunları içerir: • kalite kısıtları (örneğin kullanılabilirlik, güvenilirlik, verimlilik, taşınabilirlik) • kurumsal kısıtlar (örneğin operasyon mekanları, hedef donanım, standartlara uyumluluk) • çevresel kısıtlar (örneğin birlikte çalışabilirlik, güvenlik, mahremiyet, emniyet). • uygulama kısıtları (örneğin geliştirme dili, teslim takvimi)

KURALLAR – Kapsam
a) Bir İşlevsel Büyüklük Ölçmenin kapsamı, ölçme amacından türetilmelidir.

b) Bir ölçmenin kapsamı ölçülecek yazılımın birden fazla katmanını içermemelidir.

2.2.1 Kapsamın ölçme amacından türetilmesi

Ölçme kapsamının, ölçmeye başlamadan belirlenmesi önemlidir.

Ev inşası benzerliğine devam edecek olursak; amaç ev maliyetini kestirebilmekse evin temel, duvar, çatı gibi farklı kısımlarını farklı inşa yöntemleri gerektirdiği için ayrı ayrı ölçmek gerekebilir. Durum yazılım maliyetleri içinde aynıdır.

Geliştirilecek yazılım sistemi, sistem mimarisindeki farklı katmanlarda bulunacak parçalardan oluşuyorsa, her katmandaki yazılımın ayrı ayrı ölçülmesi gerekecektir. Yani, her parçanın büyüklük ölçme amacıyla ayrı kapsamı olacaktır. Bu Yazılım Bağlam Modeli, ilke (d)'nin sonucudur. (Katmanlar ile ilgili daha fazla bilgi için, aşağıdaki bölüm 2.2.4'e bakınız.)

Benzer biçimde, yazılım bir katman içinde farklı teknolojiler kullanan bir küme eş bileşenler olarak geliştirilecekse, ölçmeden önce her bir bileşen için ayrı bir kapsam tanımlamak gerekli olacaktır.

ÖRNEK 1: Yazılımın bileşenleri için farklı teknolojiler kullanıldığı ve ölçümlerin işgücü kestirimi amacıyla kullanılacağı durumda, her ölçüm farklı bir geliştirme üretkenliği biçimi ile ilişkilendirileceğinden her bir bileşen için farklı bir kapsam tanımlanmalıdır. (Eş bileşenler için bölüm 2.2.5'e bakınız.)

Amaç hangi yazılımın eklenip çıkarılacağına karar vermeye de yardımcı olacaktır.

ÖRNEK 2: Amaç belirli bir proje takımının geliştirdiği tüm yazılımların işlevsel büyüklüğünü ölçmekse, ilk gereken takım tarafından geliştirilmiş tüm yazılım parçalarının ve bileşenlerin İşlevsel Kullanıcı Gereksinimlerinin belirlenmesi olacaktır. Bunlar, değiştirilen yazılımdan veri çevirmek üzere bir kez kullanılmış yazılım parçasının İKG'ni içerebilir.

ÖRNEK 3: Eğer amaç yeni geliştirilmiş bir yazılımın operasyonel kullanım için uygun olan kısmının büyüklüğünü ölçmekse, veri dönüşümü için kullanılan kısım ölçme kapsamına dahil edilmeyecek ve bu büyüklük ÖRNEK 2'dekinden küçük olacaktır.

Özetle, ölçme amacı her zaman; (a) genel kapsama hangi yazılımların eklenip çıkarılacağını, (b) içerilen yazılımın, her biri kendi kapsamında ve ayrı ayrı ölçülecek biçimde, ne şekilde parçalara bölünmesi gerektiğini kararlaştırmada kullanılmalıdır.

2.2.2 Kapsam için genel örnekler

ÖRNEKLER:

- Kurumsal portföy
- Sözleşme ile anlaşılmış gereksinimlerin ifadesi
- Bir proje ekibinin teslim ettiği iş çıktısı (mevcut yazılım parametrelerini, satın alınmış paketleri, yeniden kullanılmış kodu, sadece proje için yazılmış sonradan kullanılmayan veri transfer yazılımını, yardımcı ve test yazılımlarını inceleyerek elde edilen çıktı gibi)
- Bir proje ekibinin geliştirdiği iş çıktısı (takımın geliştirdiği, sadece proje için yazılmış sonradan kullanılmayan veri transfer yazılımı, yardımcı ve test yazılımlarını içeren fakat parametre değişiklikleri, yeniden kullanılan kod ve satın alınan paketlerin sağladığı işlevselliği içermeyen çıktı gibi)
- Bir katmandaki tüm yazılımlar
- Bir uygulama
- Temel bir uygulama bileşeni
- Yeniden kullanılabilir nesne sınıfı
- Bir yazılım parçasının yeni sürümü için gerekli olan değişiklikler

2.2.3 Ayrıştırma Seviyeleri

Fark edilebileceği üzere, yukarıda listelenen bazı 'genel kapsam tipleri' aşağıda tanımlanan değişik yazılım 'ayrıştırma seviyelerine' denk gelir.

TANIM – Ayrıştırma Seviyesi
Bir yazılım parçasının bileşenlerine bölünmesi (‘Seviye 1’ gibi), bileşenlerin alt bileşenlere bölünmesi (‘Seviye 2’), daha sonra alt bileşenlerin alt-alt bileşenlere bölünmesi (‘Seviye 3’) , vb. ile elde edilen her seviye. NOT 1: ‘Tanesellik seviyesi’ ile karıştırılmamalıdır. NOT 2: Bir yazılım parçasının bileşenlerine ilişkin büyüklük ölçümleri, yalnız eş bileşenler (örneğin aynı ayrıştırma seviyesindeki bileşenler) için doğrudan karşılaştırılabilirdir.

ÖRNEK: Bir ‘uygulama portföyünün’, her biri ‘tekrar kullanılabilir nesne sınıflarından’ oluşan ‘temel (eş) bileşenlerden’ oluşan birçok ‘uygulamadan’ oluşması, farklı ‘genel kapsam tiplerine’ denk gelen farklı ayrıştırma seviyeleri için bir örnek olabilir.

Bu durumda, ölçme kapsamını belirlemek, ölçümde içerilmesi gereken işlevselliği kararlaştırmak sorusundan öte bir karardır. Karar ölçümün yapılacağı yazılımın ayrıştırma seviyelerini dikkate almayı içerebilir. Bu, farklı ayrıştırma seviyelerinde yapılan ölçümler kolayca karşılaştırılamayabileceği için, ölçme amacına bağlı olan önemli bir Ölçme Stratejisi kararıdır. Bölüm 4.3.1 kural g’de anlatılacağı üzere, bir yazılımın toplam büyüklüğü, bileşenlerinin büyüklüğü toplanarak elde edilemez.

2.2.4 Katmanlar

Ölçülecek yazılım parçasının kapsamı yalnız bir katmanla kısıtlı olacağı için, kapsam tanımlama süreci ölçenin, önce yazılım mimarisindeki katmanların neler olduğuna karar vermesini gerektirebilir. Bu sebeple bu bölümde COSMIC yönteminde kullanılan ‘katman’ ve ‘eş bileşen’ terimlerini tanımlayıp tartışacağız.

Bu tanım ve kurallara neden ihtiyaç duyulduğu şöyle açıklanabilir:

- Ölçen, yıllar içinde evrilmiş, bir mimariye uygun tasarlanmamış (spagetti mimari olarak da adlandırılır), yabancı ya da ‘miras’ kalmış ortamdaki bir yazılımı ölçmek durumunda kalabilir. Bu durumda ölçen, COSMIC terminolojisine göre katmanları ayrıştırmada rehberliğe ihtiyaç duyabilir.
- Yazılım endüstrisinde, ‘katman’, ‘katmanlı mimari’ ve ‘eş bileşen’ kavramları tutarlı şekilde kullanılmamaktadır. Ölçenin ‘katmanlı mimariyle’ sahip olduğu söylenen bir yazılımı ölçmesi durumunda, bu yazılımdaki mimari ‘katmanların’ COSMIC yöntemiyle uyumluluğunu kontrol etmesi önerilir. Bunu yapmak için ölçen, ‘katmanlı mimari’ değerler dizisindeki mimari nesnelerin, bu elkitabında tanımlanan ‘katman’ kavramı ile denkliğini kurmalıdır.

Katmanlar aşağıdaki tanımlara ve ilkelere göre belirlenebilir.

TANIM – Katman
Katman, donanımla beraber bütün bir bilgisayar sistemini oluşturan yazılım sisteminin, işlevsel parçalanması sonucu oluşan bir bölümdür. • katmanların organizasyonu hiyerarşiktir. • hiyerarşideki her seviyede yalnız bir katman vardır. • yazılım mimarisinde direkt veri alışverişinde bulunan herhangi iki katmanın işlevsel hizmetleri arasında bir ‘alt / üst’ hiyerarşi ilişkisi vardır. • yazılım mimarisinde veri alışverişinde bulunan herhangi iki katman, verinin yalnız bir kısmını aynı şekilde yorumlar.

Katman belirlemek tekrarlı bir süreçtir. Eşleştirme süreci ilerledikçe katmanlar kesin olarak ayrıştırılabilecektir. Her aday katman, belirlendikten sonra, aşağıdaki ilke ve kurallarla uymalıdır.

İLKELER – Katman

a) Bir katmandaki yazılım diğeriyle, kendi işlevsel süreçleri aracılığı ile veri alışverişinde bulunur.

b) Katmanlar arasındaki 'hiyerarşik ilişki', bir katmanın hiyerarşide, altında olan katmanların işlevsel hizmetlerini kullanacağı şekildedir. Bu şekilde kullanım ilişkilerinin olduğu durumlarda, kullanan katmanı 'üst', kullanılan katmanı 'alt' olarak tanımlıyoruz. Üst katmandaki yazılım, düzgün çalışabilmek için, alt katmanlardaki yazılım hizmetlerine güvenir; alttakiler aynı şekilde düzgün çalışabilmek için, kendi altındaki katmanlara güvenir ve bu durum, hiyerarşide aşağı indikçe devam eder. Bu durumun tersine bir alt katman, bağımlı olduğu alt katmanlarla beraber, hiyerarşide üst katmanlardaki yazılım hizmetlerine ihtiyaç duymadan çalışabilir.

c) Bir katmandaki yazılım, yazılımın alt katmanlarında sağlanan servislerin tümünü kullanmak zorunda değildir.

d) İki farklı katmandaki yazılımlar arasında alınıp verilen veri, her katmanın kendi İKG'sine göre farklı şekilde tanımlanıp yorumlanabilir. Yani, iki yazılım parçasının her biri, alınıp verilen verinin farklı veri özniteliklerini ve veri alt gruplarını tanıyabilir. Yine de, alıcı katmanın verici katmandan gelen veriyi yorumlayabilmesi için ortak olarak tanınan, alıcı yazılımın ihtiyaçlarına göre belirlenen, bir ya da daha fazla veri özniteliği ya da alt grubu olmalıdır.

KURALLAR – Katman

a) Yazılım, COSMIC modeline göre, bir katmanlar mimarisi kurgusu üzerinden tanımlanmışsa bu mimari, ölçme amacıyla katmanların belirlenmesi için kullanılmalıdır.

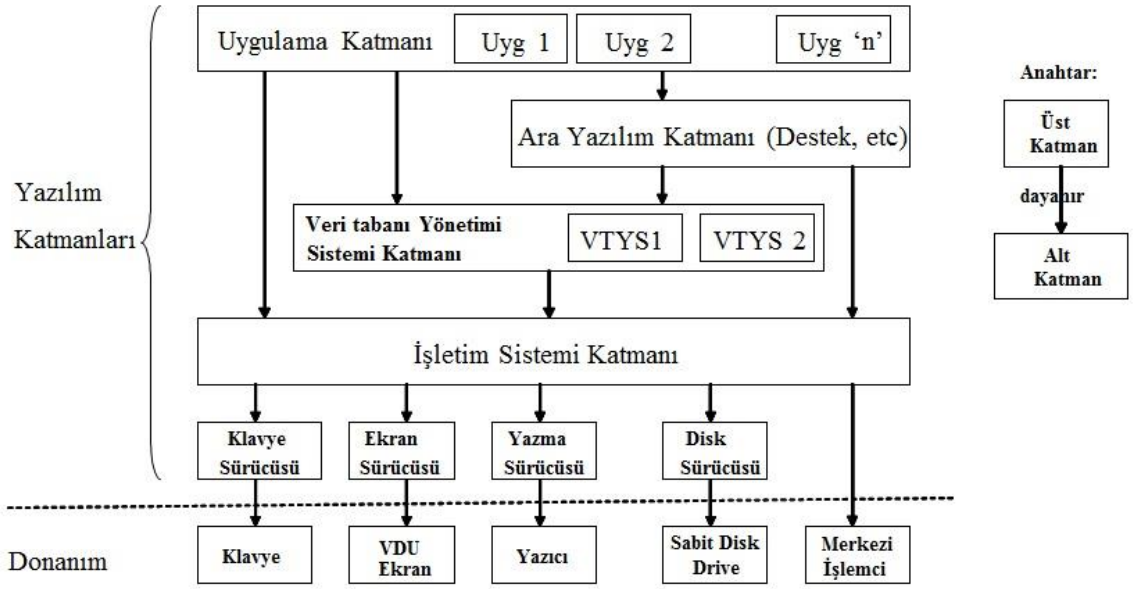
b) BYS ya da iş yazılımları alanında 'tepe' katman, başka bir katmanın altında olmayan, normalde 'uygulama' katmanı olarak adlandırılır. Bu katmandaki yazılım (uygulama) düzgün çalışabilmek için diğer tüm katmanlardaki yazılım servislerine bağlıdır. Gerçek zamanlı yazlım alanında 'tepe seviye', 'süreç kontrol yazılım sistemi' ve 'uçuş kontrol sistemi yazılımı' örneklerindeki gibi, genelde 'sistem' olarak adlandırılır.

c) Mimari tasarım ve yapı gözetilmeden evrilmiş bir yazılımın, COSMIC modeline göre katmanlara ayrılabileceğini kabul etmeyin.

Veri tabanı yönetim sistemi, işletim sistemi, aygıt sürücü gibi işlevsel hizmeti sağlayan yazılım paketleri, normalde farklı katmanlarda yer alıyormuş gibi düşünülmelidir.

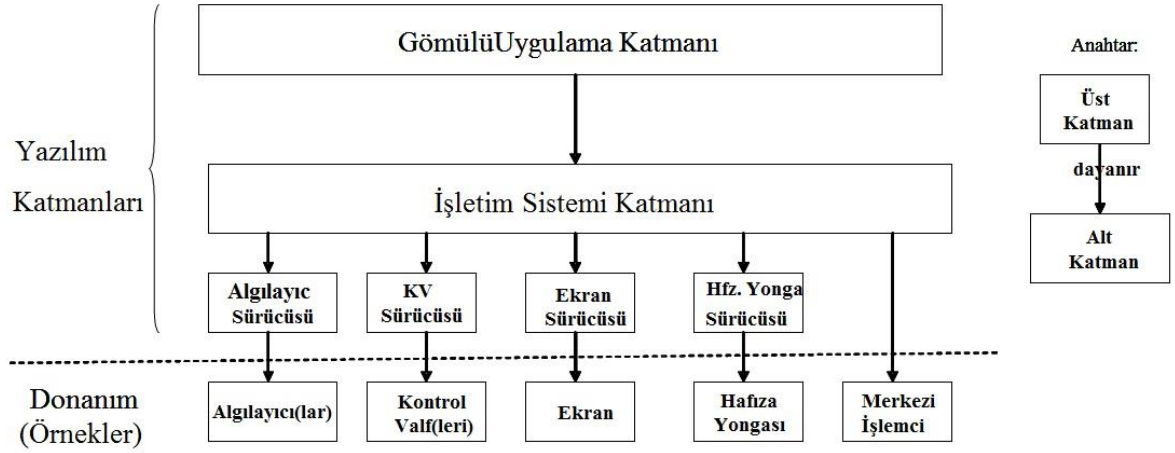
Bir kere belirlendiğinde her katman, Genel Yazılım Modeli Matrisine (Ek-A), denk gelen isimle ayrı bir 'bileşen' olarak kaydedilmelidir.

ÖRNEK 1: İş uygulamaları yazılımlarını destekleyen tipik katmanlı yazılım mimarisinin ('katman' terimini burada verdiği anlamı ile kullanan) fiziksel yapısı şekil 2.2.4.1'de verilmiştir.



Şekil 2.2.4.1 - Bir İş Uygulaması/Bilgi Sistemi için tipik katmanlı yazılım mimarisi

ÖRNEK 2: Gömülü gerçek-zamanlı yazılım parçasını destekleyen tipik katmanlı yazılım mimarisinin ('katman' terimini burada verdiği anlamı ile kullanan) fiziksel yapısı şekil 2.2.4.2'de verilmiştir.



Şekil 2.2.4.2 - Gerçek-Zamanlı Gömülü-Yazılım Sistemi için tipik katmanlı yazılım mimarisi

2.2.5 Eş bileşenler

Katman kavramının verilmesinden sonra, 'eş' şu şekilde tanımlanır:

TANIM – Eş
İki yazılım parçası aynı katmanda bulunuyorsa birbirlerinin eşidir. NOT: İki eş yazılım parçası aynı ayrıştırma seviyesinde olmak zorunda değildir.

Eş bileşenler aşağıdaki tanım ve ilkelere göre belirlenebilir.

TANIM – Eş Bileşen
Karşılıklı beraber çalışan, aynı ayrıştırma seviyesinde bulunan, bir yazılım parçasının bir katman içinde bölünmesinden oluşan ve yazılım parçasına ait İKG'nin belirli bir kısmını karşılayan bileşenlerden her biridir. NOT: Bir yazılım parçasının eş bileşenlere ayrılması, işlevsel ve/veya işlevsel olmayan gereksinimler gereği yapılabilir.

Her aday eş bileşen, belirlendikten sonra, aşağıdaki ilkelere uyumlu olmalıdır.

İLKELER – Eş Bileşen
a) Bir yazılım parçasının bir katmanında bulunan eş bileşenler kümesi arasında, katmanlarda olduğu gibi hiyerarşik bir ilişki yoktur. Bir yazılım parçasının aynı katmandaki tüm eş bileşenlerinin İKG'si, katman hiyerarşisinde aynı seviyede bulunur. b) Yazılım parçasının başarıyla çalışması için, tüm eş bileşenleri karşılıklı olarak beraber çalışmalıdır. c) Bir veri grubu, iki eş bileşen arasında; ilk bileşenin bir işlevsel sürecinin bir çıkışının, ikinci bileşenin işlevsel sürecinin bir girişi ile alınması sonucunda paylaşılabilir. Diğer bir seçenek, ilk bileşenin veri grubunu bir yazma ile kalıcı kılarak devamında, ikinci bileşenin işlevsel sürecinin bir okuması ile dolaylı yoldan paylaşması olabilir.

Bir kere belirlendiğinde her eş bileşen, Genel Yazılım Modeli matrisine (Ek-A), denk gelen isimle ayrı bir bileşen olarak kaydedilmelidir

ÖRNEK: Bir iş uygulaması 'kullanıcı arayüzü' (ya da 'önyüz'), 'iş kuralları' ve 'veri hizmetleri' bileşenleri olarak üç ana bileşen ile geliştirildiğinde, bu bileşenler eş bileşenlerdir.

Not: Aynı katmandaki iki ayrı fakat eş yazılım bileşeni veri alışverişini, ilke c) de belirtilen alternatif dizilerden herhangi biri ile gerçekleştirebilir.

Aşağıdaki şekil, örneği açıklayan bir durumu gösterir. Gösterilen tüm yazılımlar aynı katmandadır. X uygulamasının iki bileşeni ile yazılımın iki bileşeni (X uygulamasının 2 numaralı bileşeni ve Y uygulaması) arasındaki alışveriş, ilke c) de verilen alternatif serilerden herhangi biri ile gerçekleşebilir.

Şekil 2.2.5.1 – 'Eş', 'bileşen' ve 'eş bileşen' kavramları arasındaki ilişki

2.3 İşlevsel Kullanıcıların Belirlenmesi

2.3.1 İşlevsel büyüklük işlevsel kullanıcıya göre değişir

Bir benzerlik ile başlandığında, bir işyerinin zemin alanının ölçülmesi, aşağıdaki gibi dört farklı şekilde ele alınabilir (kapsam ve bahsedilen işyeri, tüm şekillerde aynıdır).

- Bina sahibi işyerinin vergilerini ödemek zorundadır. Bina sahibi için yüzey alanı, boyutlardan belirlenen ve tüm ortak koridorları, duvarların kapladığı alanları vb. içeren 'brüt metrekare' dir.
- Bina ısı yöneticisi; iç alanlar, ortak alanlar ve asansörün kapladığı alanlar gibi alanları içeren, duvar kalınlığını içermeyen net metrekare ile ilgilenir.
- İşyeri kiracısının anlaşmalı olduğu temizlik hizmetleri taşeronu ortak alanları içermeyen ama kiracının kullandığı geçiş yollarını içeren net-net artı metrekare ile ilgilenir.
- İşyeri planlama yöneticisi, sadece işyeri için kullanılabilecek net-net metrekare ile ilgilenir.

Bu benzeşimden alınacak ders şudur: Bir 'şeyin' farklı kullanıcı tipleri farklı işlevsellikler 'görebilir'; dolayısıyla, 'şeyin' farklı işlevselliklerini ölçebilir. Konu yazılım olduğunda, farklı işlevsel kullanıcılar (kullanıcı tipleri) farklı işlevselliğe ihtiyaç duyabileceği için, işlevsel büyüklükler işlevsel kullanıcı seçimine göre değişecektir.

2.3.2 İşlevsel Kullanıcılar

Bir 'kullanıcı', beklendiği gibi[5], 'ölçülen yazılımla etkileşen herhangi bir şey' olarak tanımlanır. Bu tanım COSMIC yönteminin ihtiyaçları için fazla geneldir. COSMIC yöntemi için, kullanıcı (ya da kullanıcıların) seçimi, ölçülmesi gereken işlevsel kullanıcı gereksinimleri ile belirlenir. Bu, işlevsel kullanıcı olarak bilinen, kullanıcı (tipi) aşağıdaki gibi tanımlanır.

TANIM – İşlevsel kullanıcı
Bir yazılım parçasının İşlevsel Kullanıcı Gereksinimleri verisinin göndereni yada alıcısı olan kullanıcı (tipi) .

COSMIC yönteminde ölçülecek yazılımın kullanıcılarının, diğer olası tüm kullanıcılardan ayrıştırılması gereklidir.

ÖRNEK 1: Bir iş uygulaması düşünüldüğünde işlevsel kullanıcılar, normalde insan ve arayüzü bulunan diğer eş bileşen uygulamalarından oluşacaktır. Bu şekildeki bir yazılımın işlevsel kullanıcı gereksinimleri (İKG), normalde işlevsel kullanıcılarının yazılıma veri göndereceği ya da yazılımdan veri alacağı şekilde ifade edilir.

'Yazılımla etkileşen herhangi bir şey' ifadesine göre, toplam 'kullanıcı' kümesi işletim sistemini de içermelidir. Fakat herhangi bir uygulamanın İKG'si, işletim sistemini bir kullanıcı olarak hiçbir zaman içermeyecektir. İşletim sisteminin uygulamaya getireceği kısıtlar tüm uygulamalar için geçerli olacaktır ve derleyici ya da yorumlayıcı tarafından ele alınarak uygulamanın gerçek işlevsel kullanıcılarından saklanacaktır. Pratikteki işlevsel büyüklük ölçümünde işletim sistemi, hiçbir zaman uygulamanın bir işlevsel kullanıcısı olarak düşünülmeyecektir[6].

İşlevsel kullanıcılar her zaman belirgin olmayabilir.

ÖRNEK 2: Bir mobil telefonun uygulama yazılımı örnek alınırsa (giriş kısmında bahsedilen), mobil telefonun işletim sistemini olası işlevsel kullanıcılardan çıkarmamıza rağmen, kullanıcılar hala uygulamayla direkt etkileşen tuşlara basan insanlar, donanım aygıtları (ekran, tuşlar, vb.) ve eş uygulamalar olabilir. İnsan kullanıcı, örneğin, mobil telefonun çalışabilmesi için

[5] ISO/IEC 14143/1:2007 den alınan atnım için sözlüğe bakınız.

[6] Gerçekte aslında bunun tersi doğrudur. Uygulamalar işletim sisteminin işlevsel kullanıcılarıdır. İşletim sisteminin işlevsel gereksinimleri, işlevsel kullanıcı olarak desteklenmesi gereken uygulamalar dikkate alınarak tanımlanır.

gerekli işlevselliğin sadece bir alt kümesini görecektir. Böylelikle, bu iki kullanıcı tipi farklı işlevsellikler görecektir; insan kullanıcı için İKG, telefon uygulamasının çalışması için geliştirilecek İKG'den küçük olacaktır[7].

KURALLAR – İşlevsel kullanıcılar
a) Ölçülecek bir yazılım parçasının işlevsel kullanıcıları, ölçme amacından türetilmelidir. b) Bir yazılımın ölçme amacı, yazılım parçasının geliştirilmesi ya da değiştirilmesi için gerekli olan işgücü ile ilgili olduğunda işlevsel kullanıcılar, yeni ya da değiştirilmiş işlevsellik sağlananlardır.

İşlevsel kullanıcıları tanımladıktan sonra basitçe, yazılım parçası ile işlevsel kullanıcıları arasında bulunan sınırı belirlemek kolay olacaktır. Bu alanda araya giren donanım ve yazılımları göz ardı ederiz[8].

TANIM – Sınır
Sınır, ölçülen yazılım ile işlevsel kullanıcılarının arasında, kavramsal bir arayüz olarak tanımlanır. NOT: Bir yazılım parçasının sınırı, işlevsel kullanıcıların bakış açısıyla dışarıdan algılandığı gibi, bu parça ile yazılımın çalıştığı ortam arasında kavramsal bir alandır. Sınır, ölçenin aklında karışıklığa yer vermeden, ölçülen yazılımın içinde ne olduğunu, ölçülen yazılımın çalışma ortamından ayrıştırmaya izin verir.

NOT: Bu tanım, ISO/IEC 14143/1:2007'dan alınmış olup, kullanıcıyı nitelemek üzere 'işlevsel' ön adı eklenerek değiştirilmiştir. Karışıklığa sebep vermemek için sınır, bir yazılımın etrafına, kapsamını belirlemek için çizilen bir çizgi ile karıştırılmamalıdır. Sınır, ölçme kapsamını tanımlamak için kullanılmaz.

Aşağıdaki kurallar bir 'sınır' adayını onaylamak için faydalı olabilir.

KURALLAR – Sınır
a) Ölçülen yazılım ile etkileşen işlevsel kullanıcıları belirle. Sınır, işlevsel kullanıcılar ile yazılım arasındadır. b) Tanım gereği, bir katmandaki yazılım ölçülecek diğer katmandaki yazılımın işlevsel kullanıcısı olmak üzere belirlenmiş her bir katman çifti arasında bir sınır vardır. c) Aynı katmandaki herhangi iki eş bileşeni içerek şekilde, herhangi iki yazılım parçası arasında da bir sınır vardır; bu durumda her bir bileşen, eşinin işlevsel kullanıcısı olabilir.

Sınır, bir yazılım parçasının kısmı (sınırın yazılım tarafında kalanı gibi) ile işlevsel kullanıcının kısmı olan herhangi bir şeyin (sınırın işlevsel kullanıcılar tarafında kalanı gibi) net olarak ayrımına izin verir. Kalıcı depo, yazılımın bir kullanıcısı olarak düşünülmez ve sınırın yazılım kısmında yer alır.

[7] Örneğin, Toivonen, mobil uygulamaların işlevselliğini karşılaştırmıştır.-"Defining measures for memory efficiency of the software in mobile terminals", International Workshop on Software Measurement, Magdeburg, Germany, October 2002

[8] Aslında ölçümcü veriyi gönderenler ve alanları belirlemek için İKGlerini inceletmek zorunda kaldığında, sınır zaten belirlenmiş olacaktır.

2.4 Tanesellik Seviyesinin belirlenmesi

2.4.1 Standart bir tanesellik seviyesi kullanmanın gereği

Yeni bir ev tasarımı ve yapımı için bir proje başlatıldığında, bir mimarın çizdiği ilk planlar 'üst seviyedir', yani, taslak ya da az detay halindedir. Proje yapım aşamasına doğru ilerledikçe, daha detaylı ('alt seviye') planlara ihtiyaç duyulur.

Yazılım için de aynı durum geçerlidir. Bir yazılım geliştirme projesinin başlangıç aşamalarında, İşlevsel Kullanıcı Gereksinimleri (İKG) 'üst seviyede' taslak ya da az detay halinde tanımlanmıştır. Proje ilerledikçe İKG daha 'alt seviyede' detayları gösterecek şekilde belirginleşir (1, 2, 3, vb. versiyonları gibi). Bu farklı İKG detay dereceleri farklı 'tanesellik seviyeleri' olarak bilinir. (Burada tanımlanan tanesellik ölçüsü seviyesi kavramı ile karıştırabilecek diğer terimler için kısım 2.4.3'e bakınız.)

DEFINITION – Tanesellik ölçüsü seviyesi
Bir yazılım parçasının tanımındaki, her artan genişletme seviyesinde yazılımın işlevsellik tanımı artmış ve aynı seviyede detay içermek üzere, herhangi bir genişletme seviyesi (gereksinimlerin ifadesinde, yazılımı yapısının açıklanmasında). NOT: Ölçümcüler, gereksinimler bir yazılım projesinin erken zamanlarında evrilmekteyken, herhangi bir anda gereksinim duyulan yazılım işlevselliğinin farklı parçalarının, tipik olarak, farklı tanesellik seviyelerinde dokümante edileceğinin farkında olmalıdırlar.

Ev yapım planları standart ölçütlere göre çizilir ve bir çizim üzerinde ölçülmüş bir boyutları farklı ölçütle çizilmiş bir başkasına çevirmek kolaydır. Zıt şekilde yazılımın tanımlanabileceği çeşitli tanesellik ölçüsü seviyeleri için standart bir ölçüt yoktur. Dolayısıyla farklı iki İKG ifadesinin aynı tanesellik ölçüsü seviyesinde olduğundan emin olmak zor olabilir. Ölçüm yapılacak (ya da ölçümlerin ölçeklendirileceği), standart bir tanesellik ölçüsü üzerinde anlaşmadan, iki işlevsel büyüklük ölçümünün karşılaştırılabileceğinden emin olmak imkânsızdır ve ölçümcüler bir tanesel ölçüsü seviyesindeki ölçümü diğerine ölçeklendirmek için kendi yöntemlerini geliştirmek zorunda kalabilirler.

Problemleri daha iyi gösterebilmek için bir küme yol haritasının, milli bir yol ağının detaylarını üç tanesellik ölçüsü seviyesinde gösterdiği başka bir benzeşim düşünün.

- A haritası sadece motorlu araç yollarını ve anayolları göstersin,
- B haritası tüm motorlu araç yollarını, ana ve tali yolları (motorcu atlaslarındaki gibi) göstersin,
- C haritası tüm yolları isimleri ile beraber göstersin (yerel haritalardaki gibi).

Eğer tanesellik seviyesi kavramını tanımasaydık, bu üç harita ülkenin yol ağının farklı büyüklüklerini gösteriyormuş gibi gözükecekti. Tabi ki, yol haritalarında herkes farklı seviyede detayların gösterildiğini ve her seviyede ağın büyüklüğünü anlamak üzere faklı ölçekler olduğunu anlar. Bu farklı haritaların ölçeklerinin arkasında 'tanesellik ölçüsü seviyesi' soyut kavramı vardır.

Yazılım ölçümü için, karışıklığa sebep vermeden tanımlanabilecek tekbir standart tanesellik ölçüsü seviyesi vardır. Bu bağımsız işlevsel süreçlerin belirlenip veri hareketlerinin tanımlandığı tanesellik ölçüsü seviyesidir. Ölçümler bu seviyede yapılmalı ya da mümkünse bu seviyeye ölçeklendirilmelidir[9].

[9] Bir tanesellik seviyesinden diğerine ölçümleri ölçeklendirme ile ilgili başlık Ölçüm Elkitabı bölüm 7 –"COSMIC ile Erken yaklaşık ölçme" de anlatılmıştır.

2.4.2 Tanesellik ölçüsünün netleştirilmesi

Tanesellik ölçüsünün COSMIC yöntemindeki anlamı ile ilgili karışıklık olmamasını konuda daha fazla ilerlemeden sağlamak önemlidir. Tanımlandığı üzere, bir yazılımın tanımını üst bir tanesellik ölçüsü seviyesinden daha alt bir seviyeye genişletmek, kapsamı değiştirmeden detaya girmek ile ilgilidir. Bu süreç aşağıdakilerden ile karıştırılmamalıdır.

- Farklı kullanıcılara sunulan farklı işlevsellik alt kümelerini aydınlatmak üzere yazılımı tanıtan bir ürünün detayına girilmesi ve dolayısıyla ölçülecek işlevselliği sınırlandırılması.

- Yazılımın detayına, bileşenlerinin, alt-bileşenlerinin, vb. yapısını aydınlatmak üzere detaya girilmesi (farklı 'ayrıştırma seviyelerinde' - kısım 2.2.2 bakınız). Böyle bir detaya girme ölçüm amacının tüm ölçüm kapsamının yazılımın fiziksel yapısını takip ederek alt bölümlere ayrılmasını gerektiğinde ihtiyaç duyulabilir.
- Yazılımın, geliştirme döngüsünde ilerledikçe evrilmesi, örneğin gereksinimlerden mantıksal tasarıma, fiziksel tasarıma vb. Yazılım hangi geliştirme aşamasında olursa olsun, ölçüm amacıyla sadece İKG ile ilgileniriz.

O halde, 'tanesellik ölçüsü seviyesi' kavramı sadece yazılım İKG'leri için geçerli olmak üzere yorumlanmalıdır. Diğer yazılım veya yazılım tanıtımlarının detaylarına inme ya da parçalama yolları, işlevsel büyüklüğü kullanmada ve karşılaştırmada oldukça önemli olabilir. Bu konu, "İleri ve İlgili Başlıklar, v3.0" dokümanı "Büyüklük Ölçümlerinin Karşılaştırılabilirliğinin Sağlanması" bölümünde ele alınacaktır.

2.4.3 Standart tanesellik ölçüsü seviyesi

Ölçüm için standart tanesellik ölçüsü seviyesi aşağıda tanımlandığı gibi işlevsel süreç seviyesidir[10].

TANIM – İşlevsel süreç tanesellik ölçüsü seviyesi
İşlevsel kullanıcıların • bireyler, aygıtlar yada yazlım parçaları olduğu (ve bunlardan herhangi birinin işlevsel kullanıcı grubu olmadığı) VE • yazılımın tepki vermesi gereken olayların tekil oluşumları fark edebildiği (olayların grup olarak tanımlanmadığı bir seviye) bir tanesellik ölçüsü seviyesi. NOT 1: Pratikte işlevsel kullanıcı gereksinimlerinin tanımlandığı dokümanlar, özellikle dokümanın hala evrilirken, işlevselliği farklı tanesellik ölçüsü seviyelerinde tanımlar. NOT 2: Bu işlevsel kullanıcı grupları, örneğin, üyeleri birçok işlevsel süreç tipini ele alan bir departman; yada birçok aygıt içeren bir kontrol paneli; yada merkezi bir sistem olabilir. NOT 3: Olaylar grubu, örneğin, bir İKG ifadesinde, üst seviye bir tanesellik seviyesinde ifade edilmiş, satış işlemleri olarak adlandırılmış bir muhasebe yazılım sistemi için girdiler dizisi; pilot komutları olarak bir aviyonik sisteme için girdi dizisi olabilir.

Bu tanımla beraber, aşağıdaki kural ve önerileri tanımlayabiliriz.

[10] İşlevsel süreç tanesellik seviyesi" isminin gerekçesi bu seviyenin işlevsel süreç ve veri hareketlerinin tanımlanmasıdır-işlevsel süreçlerin daha detaylı açıklaması için kısım 3.2 ye bakınız.

KURALLAR - İşlevsel süreç tanesellik ölçüsü seviyesi
a) İşlevsel büyüklük ölçümü işlevsel süreç tanesellik ölçüsü seviyesinde yapılmadır b) Henüz, tüm işlevsel süreçlerinin tanımlanmadığı, veri hareketlerinin detaylandırılmadığı bir İKG'nin işlevsel büyüklük ölçümü gerektiğinde, ölçümler tanımlanan işlevsellik üzerinden yapılıp daha sonra işlevsel süreç tanesellik ölçüsü seviyesine ölçeklendirilmelidir. (İşlevsel büyüklüğü İKG'lerini oluşturulma sürecinde erkenden kestirebilmek gibi yaklaşık büyüklük ölçme metotlarına 'İleri ve İlgili Başlıklar' dokümanından bakınız.)

Kurallara ek olarak, COSMIC, işlevsel süreç tanesellik ölçüsü seviyesinin, işlevsel büyüklük ölçümü gerekli olduğunda, kıyaslama hizmetleri sağlayıcılarına ve işlevsel büyüklük ölçme ve kullanma için tasarlanan araçlarca, örneğin proje efor kestirimi, standart olarak alınması önerir[11].

ÖRNEK: Farklı tanesellik ölçüsü seviyelerindeki işlevsellik

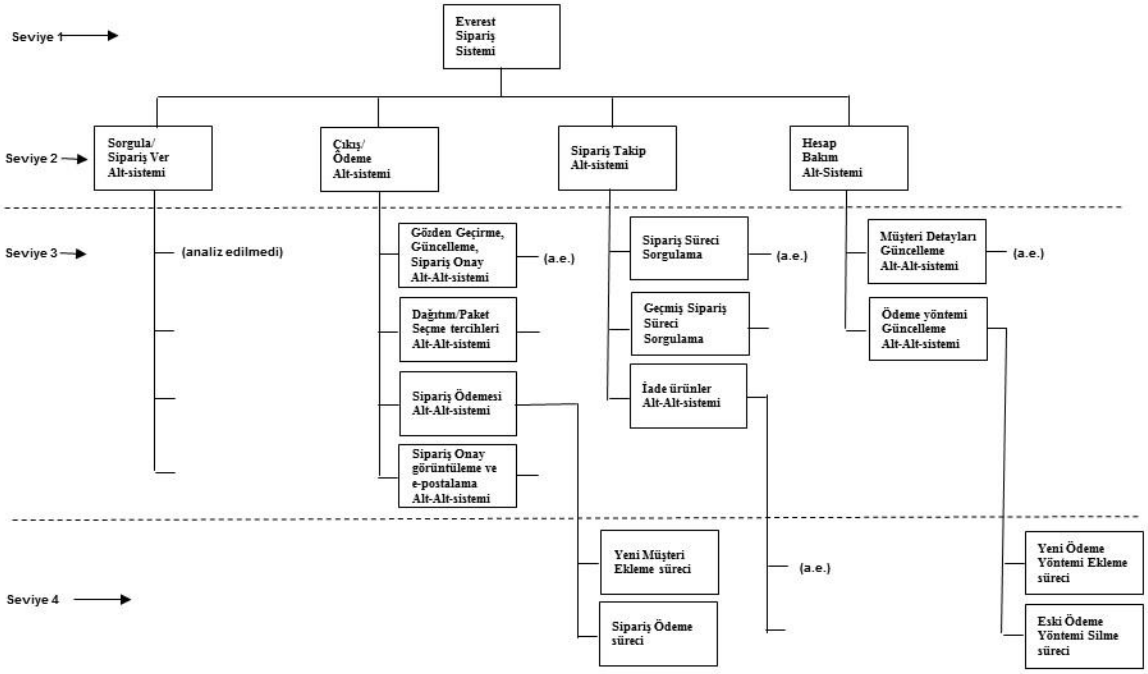
Örnek sistem, 'Everest' olarak adlandıracağımız, iş uygulamaları alanında sıkça karşılaşılan bir internet üzerinden ürün sipariş sistemidir. ('İleri ve İlgili Başlıklar' dokümanı gerçek zamanlı uygulamalar alnından bir örneği içermektedir.) Aşağıdaki tanımlama tanesellik ölçüsü seviyeleri gösterimi için basitleştirilmiştir. Tanım Everest sisteminin sadece müşterilerine sağladığı işlevselliği içermektedir. Sistemin ürünlerin sağlanmasının tamamlanması için, ürün tedarikçileri, ilan sahipleri, ödeme hizmetleri sağlayıcısı işlevsellikleri gibi, gerekli diğer işlevsellikler çıkarılmıştır.

Ölçüm kapsamı, o halde, 'Everest uygulama sisteminin müşterilere açık olan kısmı' olarak tanımlanır. Ölçüm amacını insan müşteri kullanıcılarına ('işlevsel kullanıcılar' olmak üzere) sağlanan uygulama işlevselliğin belirlenmesi olarak kabul ederiz. Uygulamanın en üstünde, 'seviye 1', Everest Sipariş Sisteminin İşlevsel Kullanıcı Gereksinimleri (İKG) ifadesi aşağıdaki gibi basit ifade olacaktır.

"Everest sistemi müşterilerin Everest'in ürün aralığında bulunan, üçüncü parti tedarikçi ürünleri de dahil olmak üzere, herhangi bir ürün için sorgulama, seçme, sipariş verme, ödeme ve teslim alma işlemlerini yapmalarına olanak sağlamalıdır."

En üst seviye ifadenin detayına girildiğinde, bir sonraki alt seviye olan seviye 2de sistemin, şekil 2.4.3.1 de gösterildiği gibi, dört alt sistemden oluştuğunu görürüz.

[11] "İşlevsel süreç tanesellik seviyesinin bir kuraldan ziyade tavsiye olarak önerilmesinin sebebi önerilenin sadece COSMIC yönteminin, in kişisel kullanıcıları iiçin değil ölçüm sonuçlarını kullanan hizmet ve araç tedarikçi ağları için de geçerli olmasıdır. COSMIC bu daha geniş kitleye sadece önerilerde bulunabilir.



Şekil 2.4.3.1 - Dört Analiz Seviyesini gösteren Everest Sipariş Sistemi kısmi analizi

Dört alt sistem şunlardır:

- Sorgu/Sipariş alt sistemi, müşterinin Everest veritabanında bir ürünü, fiyat ve elde olup olmadığını bilgileriyle bulmasını, seçilen bir ürünü almak için alışveriş sepetine eklemesini sağlar. Bu alt sistem aynı zamanda özel teklif önerileri, seçilen ürünlerin değerlendirmeleri sağlayarak satışları destekler ve teslim koşulları gibi genel sorguları sunar. Oldukça karmaşık bir alt sistem olup, örneğin amacına uygun olarak 2. Seviyenin altındaki bir seviyede daha fazla analiz edilmeyecektir.
- Çıkış/Ödeme alt sistemi, müşterinin siparişi verip, alışveriş sepetindeki ürünleri ödemesini yapmasını sağlar.
- Sipariş takip alt sistemi, siparişin teslim sürecinde hangi aşamaya geldiğini, sipariş bilgisinin kontrolünü (adres değiştirme gibi) ve ürün iadesi yapabilmesini sağlar.
- Hesap bakım alt sistemi, var olan bir müşterinin, adres ödeme biçimi gibi çeşitli hesap detaylarını güncellemesini sağlar.

Şekil 2.4.3.1 Çıkış/Ödeme, Sipariş takip ve Hesap bakım alt sistemleri üzerinde daha alt seviyelerde detaya inildiğinde ortaya çıkan detayları da gösterir. Bu detaya inme sürecinde şunlara dikkat etmek gerekir:

- Ölçülecek uygulama sisteminin kapsamını değiştirmedik
- Everest yazılımının tanımındaki tüm seviyelerde, müşterilere (işlevsel kullanıcı olarak) sağlanan işlevsellik gösterilmiştir. Bir müşteri sistemin işlevselliğini analizin bütün bu seviyelerinde görebilir.

Şekil 2.4.3.1 bu üç sistemin analizinde alt seviyelere indikçe 3. seviyede (Sipariş takip alt sistemindeki iki sorgu için) ve 4. Seviyedeki iki alt sistem için, bağımsız işlevsel süreçler bulunduğunu da gösterir. Bu örnek, işlevsellik *yukarıdan aşağıya bir yaklaşımla* analiz edildiğinde, bir 'seviyede' gösterilen işlevselliğin her zaman, COSMIC yönteminde tanımlandığı şekilde, aynı 'tanesellik ölçüsü seviyesine' denk gelmesinin kabul edilemeyeceğini gösterir. (Tanım, bir tanesellik seviyesinde işlevselliğin 'karşılaştırılabilir seviyede' olmasını gerektirir)

Dahası, diğer analistler şekli, şeklin her seviyesinde diğer işlevsellik gruplamalarını göstererek farklı biçimde çizebilir. Bunun gibi karmaşık bir sistemin analizi için tek doğru bir yol yoktur[12].

Pratikte, bu değişkenlikler kaçınılmaz olarak görüleceği düşünüldüğünde, ölçümcü ölçülecek işlevsel süreçleri bulmak için bir analiz diyagramının çeşitli seviyelerini dikkatle incelemelidir. Pratikte çoğu zaman bu, örneğin analiz henüz tüm işlevsel süreçlerin açığa çıkarıldığı aşamaya gelmediğinde, mümkün olmadığında (b) kuralı uygulanmalıdır. Bunu göstermek için Hesap Bakımı alt sistemi dalında bulunan Müşteri detayları bakımı alt alt sistemini (yukarıdaki şekil 2.4.3.1'e bakınız) inceleyelim.

Tecrübeli ölçümcü için bakım kelimesi neredeyse değişmez bir şekilde bir olaylar yani işlevsel süreçler grubu ifade eder. O taktirde, bu bakım alt alt sisteminin, müşteri detaylarını sorgula, müşteri detaylarını güncelle ve müşteri sil olmak üzere üç tane işlevsel süreci göstereceğini rahatça kabul edebiliriz.

Tecrübeli ölçümcü kabul edilen bir sayıda işlevsel süreç (burada üç tane) sayısı alıp, bunu ortalama işlevsel süreç büyüklüğü ile çarparak bu alt alt sistemin büyüklüğünü tahmin edebilir. Bu ortalama büyüklük, diğer kısımların ya da karşılaştırılabilir başka sistemlerin kalibrasyonuyla elde edilir. Kalibrasyon süreçleri ile ilgili örnekler 'İleri ve İlgili Başlıklar ' dokümanında yaklaşık büyüklük ölçümü üzerinde, yaklaşık büyüklük ölçümüne diğer yaklaşımlarla beraber verilmiştir.

Bu tip yaklaştırma yöntemlerinin açık sınırları vardır. Bu yaklaşımı yukarıda verilen İKG seviye birdeki ifadesine (Everest sistemi müşterilerin Everest'in ürün aralığında bulunan herhangi bir ürün için sorgulama, seçme, sipariş verme, ödeme ve teslim alma işlemlerini yapmalarına olanak sağlamalıdır) uyguladığımızda, az sayıda işlevsel süreç bulabiliriz. Daha detaylı analizle, bu karmaşık sistemde gerçek işlevsel süreç sayısının çok daha fazla olacağını görülecektir. Bundan dolayı, gereksinimlerim detayları belirlendikçe, kapsamda değişlik olmasa dahi, işlevsel büyüklüğün arttığı görülecektir. Sonuçta, yaklaştırma yöntemleri çok az detay bulunan üst tanesellik ölçüsü seviyelerinde dikkatle kullanılmalıdır.

'İleri ve İlgili Başlıklar' dokümanı, değişik tanesellik ölçüsü seviyelerinde büyüklük ölçümü için, telekomünikasyon yazılım mimarisinin bir parçası olan bir yazılım ile farklı tipte bir örnek verir. Bu örnek, yazılım parçasının işlevsel kullanıcıları, mimarinin diğer parçaları olduğu için 'Everest' örneğinden daha karmaşıktır. Bu örnek aynı zamanda ölçülen yazılımın ve işlevsel kullanıcılarının değişen ayrıştırım seviyeleri ile de ilgilidir

2.5 Ölçüm stratejisi evresi üzerine sonuç notları

Bir ölçüme başlamadan ölçüm stratejisi sürecinin dört elemanının dikkatli bir şekilde düşünülmesi sonuçtaki büyüklüğün doğru şekilde yorumlanabilmesini sağlayacaktır. Bu dört elaman:

a) ölçüm amacını belirle
b) ölçülecek yazılım parçasının genel kapsamın ve ayrı ayrı ölçülecek parçalarının kapsamını, yazılım mimarisinin katmanlarını ve eş bileşenleri dikkate alarak tanımla
c) ölçülecek yazılım parçasının işlevsel kullanıcılarını belirle
d) ölçülecek yazılım ürünlerinin tanesellik ölçüsü seviyesini belirle ve gerektiğinde standart işlevsel süreç tanesellik ölçüsü seviyesine nasıl ölçeklendirileceğini belirle

Gereksinimler evrildiğinde ve yeni detaylar kapsam tanımının elden geçirilmesi gerektiğini işaret ettiğinde, (b), (c) ve (d) adımlarını tekrar etmek gerekebilir.

İşlevsel büyüklük ölçümlerinin büyük bir çoğunluğu, geliştirici üretkenliğini ölçümü, proje kestirimi gibi geliştirme işgücü ile ilgili bir amaç için yapılmaktadır. Bu durumda, ölçüm stratejisini belirlemek kolay olmalıdır. Amaç ve kapsam tanımlaması kolay, işlevsel kullanıcılar geliştiricinin işlevsellik sağlayacağı kullanıcılar ve ölçümlerin yapılması gereken tanesellik ölçüsü seviyesi, işlevsel kullanıcıların tekil olayları fark edeceği seviyedir.

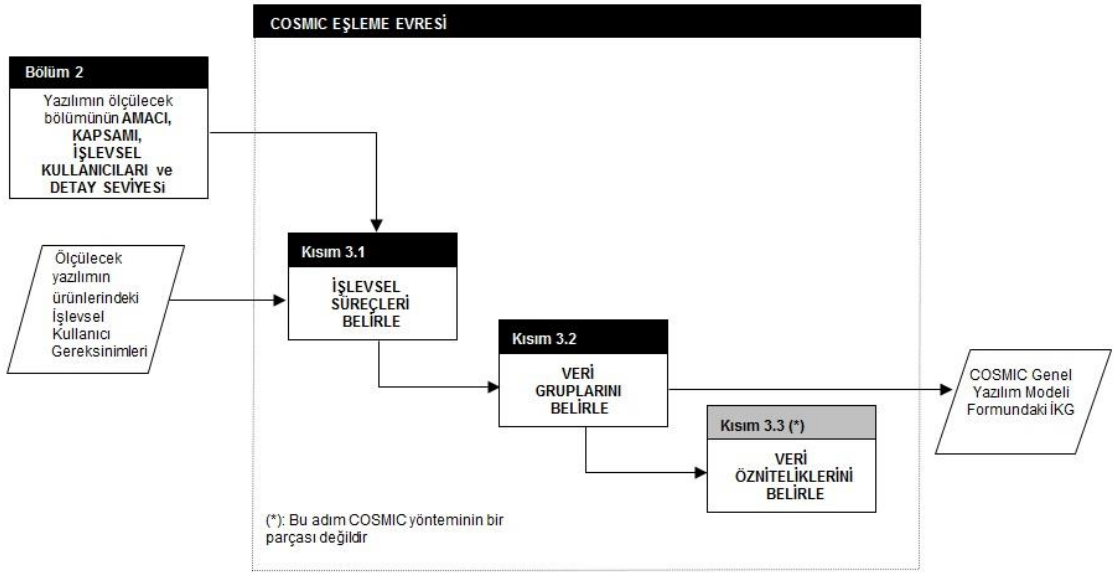
Tüm ölçümler bu kalıba uymadığı için ölçüm stratejisi her duruma uygun olarak tanımlanmalıdır.

[12] Figür 2.4.3.1 bir iyi uygulama örneği olmayabilir fakat bu tip diagramların nasıl çizilebileceği ile ilgili tipik bir örnektir.

3

EŞLEME EVRESİ

Bu bölüm eşleme sürecine ait kuralları ve yöntemleri sunmaktadır. Yazılımı COSMIC Genel Yazılım Modeline eşlemekte kullanılan genel yöntem şekil 3.0 de özetlenmektedir.



Şekil 3.0 – COSMIC eşleme süreci için genel yöntem

Bu yöntemdeki her bir adım tanımların ve kuralların bazı yol gösterici ilkeler ve örneklerle beraber sunulduğu ayrı bir bölümün (şekil 3'teki her bir adımın başlık çubuğunda belirtilmektedir) konusudur.

Yukarıda Şekil 3'te ana hatları verilen genel yöntem geniş bir aralıktaki yazılım ürünlerine uygulanabilir olacak şekilde tasarlanmıştır. Daha sistematik ve detaylı bir süreç daha özelleşmiş yazılım ürünlerinden oluşan bir grup için daha hassas eşleme kuralları sağlar ve dolayısıyla COSMIC Genel Yazılım Modeli'ni oluştururken muğlaklık seviyesini düşürür. Böyle bir süreç, tanımı dolayısıyla, ürünlerin doğasına oldukça bağımlı olacak ve bunun sonrasında da her bir organizasyonda kullanılan yazılım mühendislik yöntemlerine bağımlılaşacaktır.

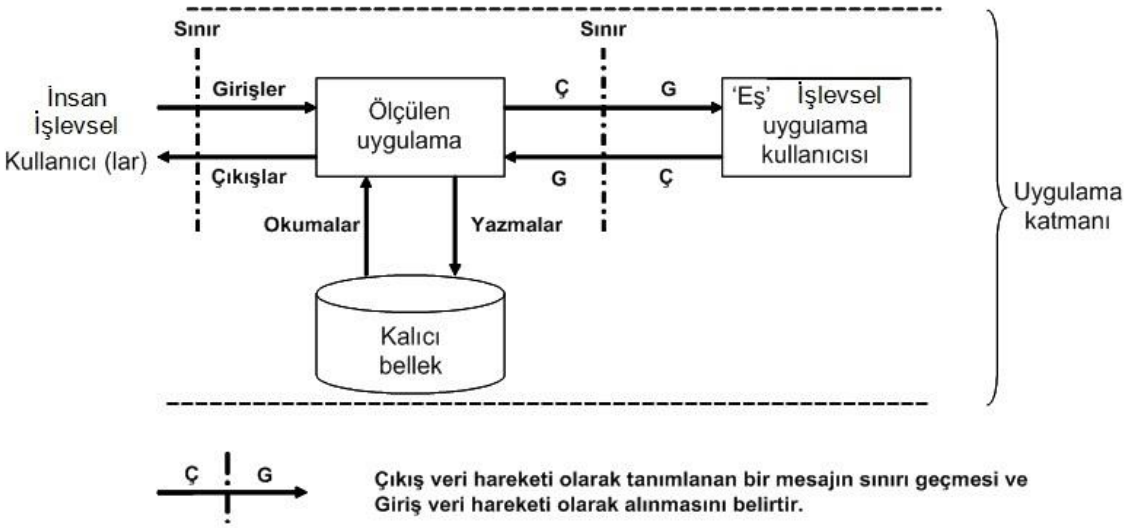
"İş Uygulamaları İçin COSMIC Ölçüm Rehberi" isimli doküman iş uygulamaları alanında kullanılan çeşitli veri analizi ve gereksinim belirleme yöntemlerinin COSMIC yöntemine eşlemesi hakkında rehberlik sunmaktadır.

3.1 Genel Yazılım Modelinin Uygulanışı

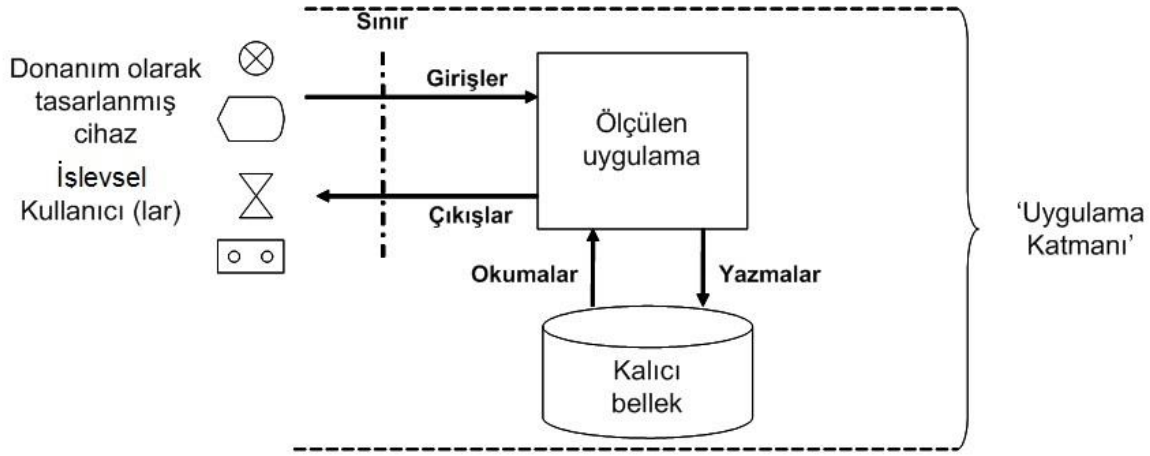
ILKELER – COSMIC Genel Yazılım Modelinin Uygulanması
COSMIC Genel Yazılım Modeli ayrı ölçme çerçevesi tanımlanan her bir ayrı yazılımın işlevsel kullanıcı gereksinimlerine uygulanacaktır. 'COSMIC Genel Yazılım Modelinin Uygulanması' demek İKG'nde belirlenen ve her bir kullanıcı (tipleri) tarafından fark edilen tetikleyici olaylar setinin belirlenmesi, ve daha sonra bu olaylara karşılık vermek üzere sağlanması gereken karşılık gelen işlevsel süreçlerin, ilgi nesnelerinin, veri gruplarının ve veri hareketlerinin belirlenmesidir.

Aşağıda verilen Şekil 3.1.1 ve 3.1.2, Yazılım Bağlam Modelinin (g) maddesindeki ilkenin uygulamasını ve Genel Yazılım Modelinin sırasıyla bir iş uygulaması yazılımının bir parçasına ve gerçek-zamanlı gömülü yazılımın tipik bir parçasına uygulanmasını gösterir.

Yukarıda verilen Şekil 2.2.3.1 ve 2.2.3.2, katmanlı yazılım mimarisinin gerçek fiziksel görünümlerini göstermesine rağmen, Şekil 3.1.1 ve 3.1.2 sırasıyla COSMIC Modellerde tanımlanmış çeşitli kavramların ilişkilerinin mantıksal bir görünümünü gösterir. Bu mantıksal görüntüde işlevsel kullanıcılar ölçülecek yazılımla bir sınır üzerindeki Giriş ve Çıkış veri hareketleri yoluyla etkileşimde bulunurlar. Yazılım veriyi Yazma ve Okuma veri hareketleri yoluyla sırasıyla kalıcı belleğe veya kalıcı bellekten hareket ettirir. Bu mantıksal görünümlerde ölçülen yazılım, işlevsel kullanıcıları ve sabit bellek arasındaki alış verişi sağlayan tüm donanım ve yazılımlar gözardı edilir.



Şekil 3.1.1 – Işlevsel kullanıcıların (mantıksal görünüm) her iki tarafta da insan ve bir 'eş' uygulama olduğu bir iş uygulaması



Şekil 3.1.2 – Donanım olarak tasarlanmış çeşitli cihazların işlevsel kullanıcı (mantıksal görünüm) oldukları bir gerçek-zamanlı gömülü yazılım uygulaması.

3.2 İşlevsel süreçlerin belirlenmesi

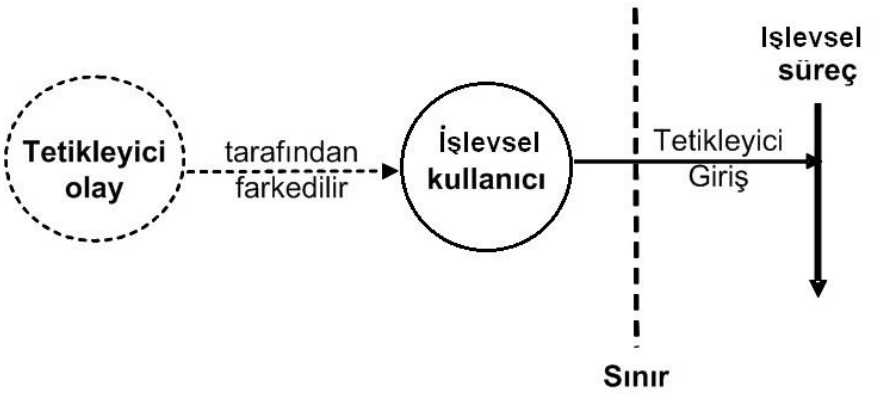
Bu adım ölçülecek yazılım parçasının, ona ait İşlevsel Kullanıcı Gereksinimlerini kullanılarak işlevsel süreç setinin belirlenmesini içerir.

3.2.1 Tanımlar

TANIM – İşlevsel Süreç
Bir işlevsel süreç benzersiz, kohesif ve bağımsız olarak çalıştırılabilen bir set veri hareketini içeren bir set İşlevsel Kullanıcı Gereksiniminin temel bileşenidir. Yazılım parçasının işlevsel olay belirlediği bilgisini veren işlevsel kullanıcılardan gelen veri hareketi (bir Giriş) ile tetiklenir. İşlevsel olaya karşılık olarak gereken işlemler yapıldığında tamamlanır. NOT: "Tetikleyici Giriş" yazılım parçasına olayın olduğunu haber vermeye ilave olarak, olayla bağlantılılı olan ilgi nesnesi hakkında veri de içerebilir.

TANIM – Tetikleyici olay
Bir yazılım parçasının bir işlevsel kullanıcısının bir veya birden fazla işlevsel süreç başlatmasına ('tetikleme') yol açan bir olay (gerçekleşen bir şey). Bir set İşlevsel Kullanıcı Gereksiniminde, işlevsel kullanıcının bir işlevsel süreç tetiklemesini sağlayan her bir olay: • adı geçen İKG seti için alt parçalara bölünemez, VE • ya olmuş veya olmamıştır. NOT: Saat ve zamanlama olayları tetikleyici olaylar olabilirler.

Bir tetikleyici olay, işlevsel kullanıcı ve bir işlevsel süreci tetikleyen Giriş verisi arasındaki ilişki aşağıda Şekil 3.2.1'de verilmektedir. Bu diyagramın yorumu şöyledir: *Bir olay bir işlevsel kullanıcı tarafından farkedilir ve işlevsel kullanıcı işlevsel bir süreci tetikler.*



Şekil 3.2.1 –Tetikleyici olay, işlevsel kullanıcı ve işlevsel süreç arasındaki ilişki

Tetikleyici Giriş, işlevsel kullanıcının tetikleyici bir olay belirlediğini yazilima haber veren normalde pozitif ve muğlak olmayan bir mesajdır. Tetikleyici Giriş genellikle olayla ilişkili ilgi nesnesi hakkında veri de içerir. Tetikleyici Giriş alındıktan sonra, bir işlevsel süreçten diğer ilgili nesneleri tanımlayan Girişleri alması ve işlemesi istenebilir.

Eğer bir işlevsel kullanıcı doğru olmayan bir veri yollarsa, örneğin bir algılayıcı-kullanıcı hatalı çalışıyorsa veya bir insan tarafından girilen sırada hata varsa, olayın gerçekten olup olmadığını ve/veya girilen verinin gerçekten geçerliliği ve nasıl cevap verileceğinin belirlenmesi genellikle işlevsel sürecin görevidir.

3.2.2 işlevsel süreçlerin belirlenmesi yaklaşımı.

İşlevsel süreçlerin belirlenmesi için olan yaklaşım ölçümcüye sağlanan yazılım ürünlerine, yani ölçümün gerektiği anki yazılım yaşam döngüsü noktasına ve kullanılan yazılım analiz, tasarım ve geliştirme yöntemlerine bağlıdır. Metodolojilerin belirli bir yazılım alanında bile büyük oranda değişebileceğini göz önüne alarak, işlevsel gereksinimleri tanımlamak için bir veya daha fazla sayıda genel süreç önermek bu Ölçüm El Kitabının çerçevesinin ötesindedir.

"İş Uygulamaları İçin COSMIC Ölçüm Rehberi" dokümanı, kısım 4.4 iş uygulamaları alanında işlevsel süreçlerin nasıl belirleneceği ve seçileceği konusunda daha fazla kural ve örnekler sunmaktadır.

En önemli genel tavsiye, her bir ayrı olay (-tipi) genellikle bir (fakat bazen birden fazla) işlevsel sürece (-tipine) yol açtığından işlevsel kullanıcıların dünyasında yer alan, yazılımın cevap vermesi gereken ayrı olayların belirlenmeye çalışılmasının neredeyse her zaman kullanışlı olduğudur. Yazılımın ilgilenmesi gereken her geçiş bir olaya karşılık geldiğinden, olaylar durum diyagramlarında ve varlık yaşam döngüsü diyagramlarında tanımlanabilirler.

Aday işlevsel süreçlerin düzgün olarak belirlendiğinin kontrol edilmesi için aşağıdaki kurallar kullanılmalıdır.

KURALLAR – İşlevsel süreç
a) Bir işlevsel süreç anlaşılan kapsamdaki en az bir belirlenebilir İşlevsel Kullanıcı Gereksiniminden türetilecektir.
b) Bir işlevsel süreç (-tipi) belirlenebilir bir tetikleyici olay (-tipi) olunca gerçekleştirilecektir.
c) Belli bir olay (-tipi) paralel olarak çalışan bir veya birden fazla işlevsel süreç (-tipi) tetikleyebilir. Belli bir işlevsel süreç (-tipi) bir veya birden fazla olay (-tipi) tarafından

tetiklenebilir.

d) Bir işlevsel süreç en azından iki veri hareketi içerecektir, bir Giriş ve bir Çıkış veya Yazdırma.

e) Bir işlevsel süreç sadece ve sadece bir yazılım katmanına ait olacaktır.

f) Gerçek-zamanlı yazılım bağlamında, bir işlevsel süreç kendi tarafından oluşturulan bir bekleme durumuna girdiğinde (yani işlevsel süreç tetikleyici olaya karşı yapması gereken herşeyi yapıp bir sonraki tetikleyici Girişi alana kadar beklemeye başladığında) sonlanmış kabul edilecektir.

g) Ait olduğu İKG bir işlevsel sürecin maksimum sayıdaki girdi veri özniteliklerinden oluşan değişik alt-kümeleriyle oluşmasına izin verse bile ve bu değişimler ve/veya değişen girdi değerleri işlevsel süreçte değişik işlemsel patikalara yol açsa bile bir işlevsel süreç (-tipi) tanımlanacaktır.

h) Aşağıdaki durumlarda ayrı olay (-tipleri) ve bunun sonucunda da ayrı işlevsel süreç (-tipleri) ayırt edilmelidir:

- Kararlar zaman içinde salıverilen ayrı olaylara yol açtığında, (örneğin, emir verisini bugün girmek ve daha sonra emrin kabulünü onaylamak ayrı bir karar gerektirdiğinden iki ayrı işlevsel sürecin belirtilmesi olarak değerlendirilmelidir),
- Aktiviteler için sorumluluklar ayrıldığında (örneğin, bir personel sisteminde temel kişisel verilerin idamesinin sorumluluğunun bordro verilerinin idamesinin sorumluluğundan ayrılmasının ayrı işlevsel süreçleri belirtmesi; veya geliştirilmiş bir yazılım paketinde sistem yöneticisi için 'normal' kullanıcılara sağlana işlevsellikten ayrı olarak paketin parametrelerini idame ettirme işlevselliğinin olması.)

3.2.3 İş uygulamaları alanında tetikleyici olaylar ve işlevsel süreçler

a) Çevirimiçi bir iş uygulamasının tetikleyici olayları genellikle uygulamanın insan işlevsel kullanıcıları gerçek dünyasında olur. İnsan kullanıcı olayın oluşmasını olay hakkında veri girerek işlevsel sürece aktarır.

ÖRNEK 1: Bir şirkette bir sipariş alınır (tetikleyici olay), bir çalışanın (işlevsel kullanıcı) emir verisini 'emir girme' işlevsel sürecinin ilk veri hareketi olarak girmesine yol açar (tetikleyici Giriş in ilgi nesnesi olan "emir" hakkında veri aktarması).

b) Bazen bir A uygulamasının eş uygulaması olan B uygulamasının A uygulamasına veri yollaması veya ondan veri alması ihtiyacı vardır. Bu durumda, eğer B uygulaması A uygulamasına veri göndermek veya ondan veri almak ihtiyacındaysa, B uygulaması A uygulamasının işlevsel bir kullanıcısıdır.

ÖRNEK 2: Diyelim ki örnek 1) de verilen emrin alınmasında emir uygulamasının müşteri detaylarını ölçülmekte olan merkezi bir müşteri-kayıt uygulamasına yollaması gerektiğini varsayın. Bu durumda emir uygulaması merkezi uygulamanın işlevsel kullanıcısı durumuna geçmiştir. Emir uygulaması müşteri verisinin alınışını hisseder ve merkezi uygulamaya tetikleyici bir Girdi olarak ilgi nesnesi 'müşteri' hakkında veri yollayarak merkezi müşteri-kayıt uygulamasında bu verinin saklanması için bir işlevsel süreç tetikler.

c) Temel olarak bir işlevsel sürecin analizinin çevirimiçi olarak mı veya toplu işlem modunda mı yapılacağı fark etmez. Geceden sabaha işleme teknik bir uygulama kararıdır ve geceden sabaha işlem tamamlanana kadar 'yapılması gerekli her şey yapıldı' kararı oluşmaz. Bir toplu-işleme akışı herbiri aynı akış içerisindeki işlevsel süreçlerden bağımsız olarak uç uca analiz edilmesi gereken bir veya birden fazla işlevsel süreç (-tipleri) içerir. Her bir işlevsel süreç kendisine ait ölçülmesi gereken tetikleyici bir Giriş'e sahiptir.

ÖRNEK 3: Diyelim ki örnek 1 de verilen emrin çevirimiçi olarak girildiğini, fakat otomatik olarak toplu olarak geceden sabaha işleme ile yapılmak üzere saklandığını varsayın. İşlevsel kullanıcı hala emirleri çevirimiçi olarak giren insandır; tetikleyici Giriş hala emir verisidir. Emirlerin girişi ve işlenmesi için bir işlevsel süreç vardır.

d) Bir saatten gelen periyodik sinyaller (saat vuruşları) bir işlevsel süreci fiziksel olarak tetikleyebilir.

ÖRNEK 4: Bir işletmenin yılsonu durumunu raporlayan ve gelecek yıl için pozisyonları sıfırlayan bir yıl sonu toplu işlem İKG'si olduğunu varsayın.Fiziksel olarak işletim sistemi tarafından oluşturulan bir yıl sonu saat vuruşu bir veya birden fazla işlevsel süreci içeren toplu akışın başlamasına yol açar. Akıştaki giriş verilerini kullanan akış içerisindeki bu işlevsel süreçler normal şekilde analiz edilmelidir (örneğin, herhangi bir işlevsel süreç için girdi verileri, ilki o süreç için tetikleyici Giriş olan bir veya daha fazla Giriş'I içerir).

Bununla beraber, akış içerisinde raporlanını oluşturmak için herhangi bir girdi verisine ihtiyaç göstermeyen bir işlevsel süreç olduğunu varsayın. Fiziksel olarak, (insan) işlevsel kullanıcı bu işlevsel sürecin tetiklenmesini işletim sistemine devretmiştir.Her bir işlevsel sürecin bir tetikleyici Giriş'i olması zorunlu olduğundan, yıl sonu toplu işlemini başlatan yıl sonu saat vuruşu bu süreç için bu rolü üstlenmiş olarak kabul edebiliriz. Bu işlevsel süreç raporlarını oluşturmak için birden çok Okuma ve Çıkış!a ihtiyaç gösterebilir. Mantıksal olarak, bu örnekteki analiz, kullanıcının rapor üretilmesini toplu akış, ile işletim sistemine devretmek yerine çevirimiçi menu birimlerinden bir fare tıkı ile bir veya birden fazla raporun üretilmesini başlatmasından farklı değildir.

e) Tek bir olay bağımsız olarak çalışan bir veya birden fazla işlevsel süreci tetikleyebilir.

ÖRNEK 5: Hafta sonunun başlangıcını belirten bir saat vuruşu üretim raporlarının başlamasına ve sürecin bir iş akışı sistemindeki zaman sınırlarının bitişini gözden geçirmesine yol açar.

f) Tek bir işlevsel süreç bir veya birden fazla tipteki tetikleyici olay tarafından tetiklenebilir.

ÖRNEK 6: Bir bankacılık sisteminde, tüm bir hesap özeti bir ay sonu toplu işlemiyle tetiklenebileceği gibi, belirli bir müşteri isteğiyle de tetiklenebilir.

Toplu akışlarda tetikleyici olayları ve işlevsel süreçleri ayırma konusunda çeşitli örnekler için bkz. '

"İş Uygulamaları İçin COSMIC Ölçüm Rehberi" , kısım 4.6.3.

3.2.4 Gerçek-zamanlı uygulama alanından tetikleyici olaylar ve işlevsel süreçler

a) Tetikleyici bir olay tipik olarak bir algılayıcı tarafından algılanır.

ÖRNEK 1: Sıcaklık belirli bir değere ulaştığında (tetikleyici olay), bir algılayıcının bir uyarı ışığının yanması (işlevsel süreç) için bir sinyal yollaması (tetikleyici (Giriş veri hareketi) sağlanır.

ÖRNEK 2: Askeri bir uçakta "füze yaklaşıyor" olayını tespit eden bir algılayıcı bulunmaktadır. Algılayıcı tehlikeye karşılık vermesi gereken gömülü yazılımın bir işlevsel kullanıcısıdır. Bu yazılım için, bir olay algılayıcı bir şey tespit ettiği zaman gerçekleşir ve yazılımı ona bir mesaj (tetikleyici Giriş) yollayarak tetikleyen algılayıcıdır (işlevsel kullanıcı), örneğin 'algılayıcı 2 bir füze tespit etti' gibi bir mesaj ve ilave olarak füzenin ne hızla yaklaştığını ve koordinatlarını belirten bir veri akışı gibi. İlgi nesnesi füzedir.

b) Bir saatten gelen periyodik sinyaller (saat vuruşları) bir işlevsel süreci tetikleyebilir.

ÖRNEK 3: Bazı gerçek zamanlı süreç kontrol yazılımlarında bir saatin (işlevsel kullanıcı) vuruşu (tetikleyici olay) saatin işlevsel sürece normal kontrol döngüsünü yinelemesini belirten bir bit'lik veri mesajı içeren bir sinyal (tetikleyici Giriş) yollamasına yol açar. Bundan sonra işlevsel süreç çeşitli algılayıcıları okuyarak ilgi nesneleri hakkında veri alır ve gerekli işlemleri yapar. Saat-vuruşuna eşlik eden başka bir veri yoktur.

c) Tek bir olay bağımsız ve paralel olarak çalışan bir veya birden fazla işlevsel süreci tetikleyebilir.

ÖRNEK 4: Bir nükleer güç santralında tespit edilen bir acil durum santralin değişik bölgelerinde kontrol çubuklarının alçaltılması, acil soğutma başlatılması, vanaların kapatılması, kullanıcıların uyarılması için alarmları çalmak vb. işlevsel süreçleri tetikleyebilir.

d) Tek bir işlevsel süreç bir veya birden fazla tipteki tetikleyici olay tarafından tetiklenebilir.

ÖRNEK 5: Bir uçağı tekerleklerinin kaldırılması "ağırlık-yerden-kalktı" algılayıcısı veya pilot komutuyla tetiklenebilir.

3.2.5 Ayrık işlevsel süreçler üzerine Ek Bilgiler

Yazılım olayları ayrıştırır ve bunlara karşılık gelen işlevsel süreçlerini sadece İKG'sine dayanarak sağlar. Yazılımı boyutlandırırken bazen yazılımın tanıması gereken ayrı olayların ne olduğuna karar vermek zor olabilir. Bu genellikle özgün İKG'nin artık ortada olmadığı ve örneğin kullanıcının ekonomik bulduğu için birden fazla gereksinimi birleştirdiği durumda gerçekleşir. Veri girdilerinin organizasyonunu incelemek (aşağıya bakınız) veya yüklenen yazılım için menüleri incelemek yazılımın cevap vermesi gereken ayrı olayların belirlenmesi ve ilintili işlevsel süreçlerin analizinde yardımcı olabilir.

ÖRNEK 1: İlave bir çocuk için vergi indirimi ve aynı zamanda düşük gelirliler için 'çalışan vergi indirimi' için birer kullanıcı gereksinimi varsa bunlar insan kullanıcılar dünyasında yazılımın cevap vermesi gereken ayrı olaylara karşılık gelen gereksinimlerdir. Her iki durum için de veriyi edinmek için tek bir vergi formu kullanılsa da, sonuç olarak iki işlevsel süreç olmalıdır.

ÖRNEK 2: Bir durumda (bir kişi için) gelir miktarı çalışan vergi indirimi için olan limiti aşarsa ve başka bir durumda (başka bir kişi için) aşmıyorsa, bu farklılık iki ayrı işlevsel sürece yol açmaz, fakat aynı süreç içinde iki farklı durumun sağlandığını gösterir.

ÖRNEK 3: Kural (g)'ye örnek olarak, eğer belirli bir olay A, B ve C veri özniteliklerini içeren bir veri grubunun Giriş'ini tetikliyorsa, ve İKG aynı olayın başka bir anının A ve B öznitelikleri için değer taşıyan bir veri grubunun Giriş'ini tetiklemesine izin veriyorsa, bu iki işlevsel sürecin (-tipinin) tanımlanmasına yol açmaz. Veri öznitelikleri A,B ve C'yi hareket ettiren ve değiştiren sadece bir Giriş ve bir işlevsel süreç (-tipi) belirlenmiştir.

Bir kez belirlendikten sonra, her bir işlevsel süreç Genel Yazılım Modeli matrisinde (ek A), uygun katman veya eş bileşen altında, karşılık gelen etiketle kendisine ait bir satırda kayıt edilebilir.

3.2.6 Eş Bileşenlerin İşlevsel Süreçleri

Ölçümün amacı her bir eş bileşeni ayrı ayrı ölçmek olduğunda, her bir eş bileşen için farklı ölçüm çerçeveleri tanımlanmalıdır. Bu durumda eş bileşenlerin işlevsel süreçlerinin büyüklükleri daha önceki bölümlerde tanımlanmış kurallarla ölçülür.

Yazılım iki veya daha çok eş bileşen içeriyorsa, her ölçüm süreci için (...kapsam tanımlama, işlevsel kullanıcı ve çerçeve tanımlama..vb), ölçüm kapsamları örtüşmemelidir. Her bileşenin ölçüm kapsamı tüm işlevsel süreç setini tanımlamalıdır. Örneğin bir parçası bir eş bileşenin, diğer parçası diğer eş bileşenin kapsamında bulunan bir işlevsel süreç olamaz. Aynı şekilde, bir eş bileşenin ölçüm kapsamında bulunan işlevsel süreç, mesaj değişimi yapsalar dahi diğer eş bileşenin ölçüm kapsamında bulunan bir işlevsel süreç hakkında bilgi sahibi değildir.

Her bileşenin işlevsel kullanıcıları, incelenen bileşen içerisindeki işlevsel süreçleri tetikleyen olaylar incelenerek belirlenir. (Tetikleyici olaylar sadece işlevsel kullanıcıların dünyasında gerçekleşebilir.)

Şekil 4.1.8.2 iki eş bileşenin işlevsel süreçlerini ve veri hareketlerini göstermektedir.

3.3. İlgi alanındaki nesnelerin ve veri gruplanın belirlenmesi

3.3.1 Tanımlar ve Ilkeler

Bu adım ölçülecek yazılım parçası tarafından referans verilen veri gruplarının tanımlanmasını içerir. Özellikle iş uygulamaları alanında veri gruplarını belirlemek için genellikle ilgi nesnelerini ve olası olarak onların özniteliklerini belirlemek yardımcı olabilir. Veri grupları bir sonraki kısımda anlatılacak olan 'veri hareketlerine' kaydırılır.

TANIM – İlgi nesnesi
İşlevsel Kullanıcı Gereksinimleri bakış açısından belirlenen herhangi bir 'şey'. Fiziksel bir şey olabileceği gibi, işlevsel kullanıcı dünyasında yazılımın veriyi işlemesi ve/veya saklaması gereken kavramsal bir nesne veya bir kavramsal nesnenin parçası olabilir. NOT: COSMIC yönteminde, 'ilgi nesnesi' ifadesi diğer yazılım mühendisliği ile ilgili

ifadelerden kaçınmak içindir. Bu ifade Nesne Yönelimli yöntemlerde kullanıldığı şekliyle 'nesne'leri vurgulamaz.

TANIM – Veri grubu

Bir veri grubu her bir veri özniteliğinin aynı ilgi nesnesinin tamamlayıcı özelliklerini tanımladığı farklı, boş olmayan, sıralı olmayan ve tekrarlamayan bir veri öznitelikleri setidir.

TANIM – Kalıcı bellek

Kalıcı bellek bir işlevsel sürecin bir veri grubunu işlevsel sürecin ömrünün ötesinde depolamasına olanak tanıyan ve/veya bir işlevsel sürecin kendisinden başka bir işlevsel sürecin veya kendisinin daha önceki bir anının depoladığı veri grubunu geri almasına olanak tanıyan bellektir..

NOT 1: Kalıcı bellek sınırın yazılım tarafında olduğu için, COSMIC modelde yazılımın bir kullanıcısı olarak değerlendirilmez.

NOT 2: 'Bir diğer süreç' e örnek salt-okunabilir belleğin üretimi olabilir.

Belirlendikten sonra, her bir aday veri grubu aşağıdaki ilkelerle uyumlu olmalıdır:

İLKELER – Veri grubu.

a) Her bir belirlenen veri grubu kendisinin tek olan veri öznitelikleri topluluğu yoluyla tek ve ayrılabilir olacaktır.
b) Her bir veri grubu yazılımın Işlevsel Kullanıcı Gereksinimlerindeki herhangi bir ilgi nesnesi ile direk olarak ilişkili olacaktır.
c) Bir veri grubu yazılımı destekleyen bir bilgisayar sisteminde gerçekleştirilecektir.

Belirlendikten sonra, her bir veri grubu Genel Yazılım Modeli matrisindeki (ek A) karşılık gelen etiket altında bireysel bir kolona yazılabilir.

3.3.2 Bir veri grubunun gerçekleştirilmesi hakkında

Pratikte, veri gruplarının gerçekleştirilmesi çeşitli şekillerde olabilir, örneğin:

a) Kalıcı bellek cihazında fiziksel bir kayıt yapısı olarak (dosya, veri tabanı, salt-okunabilir bellek vb.).
b) Bilgisayarın kısa süreli belleğinde fiziksel bir yapı olarak (dinamik olarak atanmış veri yapısı veya önden atanmış bir bellek alanı bloğu).
c) Bir girdi/çıktı cihazında sunulan (görüntü ekranı, basılı rapor, kontrol panel göstergesi, vb.) işlevsel olarak ilintili kümelenmiş veri öznitelikleri.
d) Bir cihaz ve bilgisayar arasında veya bir ağ üzerinden mesaj iletimi olarak.

3.3.3 İlgi nesnelerinin ve veri gruplarının belirlenmesi hakkında

İlgi nesnesi ve veri grupları için verilen tanım ve ilkeler olası en geniş yazılım alanınına uygun olması amacıyla kasıtlı olarak geniş verilmiştir. Bu kalite bazen belirli bir yazılım parçasının ölçülmesi sırasında tanım ve ilkelerin uygulanmasını güçleştirmesi sonucuna yol açmaktadır. Bu yüzden belirli

durumlar için ilkelerin uygulanmasına yardımcı olmak amacıyla aşağıdaki durumlar ve örnekler hazırlanmıştır.

Bir işlevsel sürecin içine veya dışına hareket ettirilen veya bir işlevsel süreç tarafından kalıcı belleğe doğru veya kalıcı bellekten hareket ettirilen bir grup veri özniteliklerini analiz etme ihtiyacıyla karşılaşıldığında, tüm özniteliklerin tek bir 'ilgi nesnesi' hakkında bilgi taşıması, COSMIC yönteminde tanımlandığı şekilde ayrı 'veri grupları'nın sayısını belirleyen faktör olduğundan, kritik derecede önemlidir. Örneğin, eğer bir işlevsel sürece girdi olan veri öznitelikleri üç ayrı ilgi nesnesinin öznitelikleri ise, üç ayrı 'Giriş' veri hareketi tanımlamamız gerekir.

İş uygulamaları alanında ilgi nesnesi ve veri grupları

ÖRNEK 1: İş uygulamaları yazılımları alanında, yazılımın çalışanlar veya emirlerle i veri saklaması gerekliliğini varsayarak, bir ilgi nesnesi bir 'çalışan' (fiziksel) veya bir 'emir' (kavramsal) olabilir. 'Emir' durumunda genellikle çok-satırlı emirlerin İKG'sine bakarak iki ilgi nesnesi belirlenir: 'emir' ve 'emir-satırı'. Karşılık gelen veri grupları 'emir verisi' ve 'emir satır verisi' olarak adlandırılabilirler.

Veri grupları kalıcı bellekte tutulmayan fakat kalıcı bellekte tutulan veriler ile türetilebilen bir 'şey hakkında veri isteyen' herhangi bir rasgele sorgulama yapıldığında oluşturulurlar.Bir rasgele sorgudaki Giriş veri hareketleri (gerekli veriyi türetmekteki seçim parametreleri) ve Çıkış veri hareketlerinin (istenilen öznitelikleri içeren) her ikisi de bu 'şey' hakkındaki veri gruplarını hareket ettirirler. Bunlar işlevsel sürecin çalışması süresince yaşamlarını sürdüremeyen geçici veri gruplarıdır. Yazılım ve kullanıcısı(ları) arasındaki sınırı geçtikleri için geçerli veri gruplarıdır.

ÖRNEK 2: Bir personel veri tabanında 35 yaşın üzerindeki tüm çalışanların isimlerini listeleyen bir rasgele sorgulama vardır. Giriş seçim parametrelerini içeren bir veri grubunu hareket ettirir. Çıkış tek öznitelik olarak 'isim'i içeren veri grubunu hareket ettirir; 'ilgi nesnesi' (veya 'şey') ise '35 yaşın üzerindeki tüm çalışanlar'dır. İşlevsel süreçleri kayıt ederken geçici bir veri grubunu rasgele sorgunun sonuçlarının türetildiği ilgi nesnesi(leri) ile ilişkilendirmek yerine ilgili olduğu ilgi nesnesine göre isimlendirmek önemlidir.

İlgi nesnelerini ve ayrı veri gruplarını belirlemek için kullanılan veri analizi yöntemleri hakkında detaylı tartışmalar için bkz. "İş Uygulamaları İçin COSMIC Ölçüm Rehberi" .

Gerçek zamanlı yazılım alanında ilgi nesnesi ve veri grupları

ÖRNEK 3: Fiziksel cihazlardan bir Giriş olan tipik olarak bir tek ilgi nesnesinin durumu hakkında, örneğin bir vananın açık/kapalı olması, kısa süreli sabit bellekteki bir verinin geçerli/geçersiz olacağını belirten bir zaman verisi gibi, veya kritik bir olayın olduğunu belirten ve bir kesmeye yol açan tipteki verileri içeren veri hareketleri. Benzer olarak bir uyarı lambasını açıp kapamaya yarayan bir Çıkış komutu da tek bir ilgi nesnesi hakkında veri taşır.

ÖRNEK 4: Bir mesaj anahtarı bir mesaj veri grubunu Giriş olarak alabilir ve ait olduğu yazılım parçasının İKG'si uyarınca hiç değiştirmeden Çıkış olarak yönlendirebilir.Mesaj veri grubunun öznitelikleri örneğin, 'yollayan, alan, yönlendirme_kodu, ve mesaj_içeriği' olabilir; mesajın ilgi nesnesi ise 'mesaj'dır.

ÖRNEK 5: Işlevsel Kullanıcı Gereksinimlerinde belirtilen ilgi nesnelerini temsil eden, işlevsel süreçlerle sürdürülen ve ölçülecek yazılımda yer alan işlevsel süreçler tarafından ulaşılabilir olan ortak bir veri yapısı.

ÖRNEK 6: Işlevsel Kullanıcı Gereksinimlerinde bulunan ve kalıcı bellekte tutulan (örneğin ROM) grafik ve tabloların değerlerini temsil eden ve ölçülecek yazılımda yer alan işlevsel süreçler tarafından ulaşılabilir olan referans bir veri yapısı.

ÖRNEK 7: Genellikle 'düz dosyalar' olarak tanımlanan, İşlevsel Kullanıcı Gereksinimlerinde ifade edilen ilgi nesnelerini temsil eden, kalıcı bellek aygıtlarında tutulan dosyalar.

3.3.4 Veri hareketleri için aday olmayan veri veya veri grupları

Bir işlevsel kullanıcıya ait bir ilgi nesnesiyle ilişkili olmayan, dolayısıyla ölçülmeyecek olan, fakat giriş ve çıkış ekranları veya raporlarında yer alan herhangi bir veri, veri hareketi yapacak şekilde tanımlanmamalıdır.

ÖRNEK 1: Tüm ekranlarda görülen, örneğin sayfa başlığı, sayfa altlığı gibi genel uygulama verileri (şirket ismi, uygulama ismi, sistem günü, vb.).

ÖRNEK 2: İşlevsel kullanıcının veriyi hareket ettirmek yerine yazılımın kullanımını kontrol ettikleri kontrol komutları (sadece iş uygulamaları sahasında tanımlanan bir kavram), örneğin sayfa yukarı/aşağı komutları, bir hata mesajını onaylamak için OK tuşuna basmak vb. Bkz. Kısım 4.1.10.

COSMIC Genel Yazılım Modeli bir işlevsel süreç içerisindeki tüm veri kullanımının dört veri hareketi tipinden biriyle ilişkili olduğunu varsayar, bkz. kısım 4.1.6. Bu yüzden işlevsel bir süreç içerisinde Giriş, Çıkış, Okuma ve Yazma ya ilave olarak verinin hiçbir hareketi veya kullanımı veri hareketi adayı olarak tanımlanamaz. (Veri hareketleri ve kullanımının yanlış yorumlanabildiği durumlara örnek olarak Okuma için kısım 4.1.4, madde c) ve Yazma için ise kısım 4.1.5, madde d) ye bakınız).

3.3.5 İlgi nesnesi olarak işlevsel kullanıcı

Bir çok basit gerçek zamanlı yazılımlarda, kısım 3.3.3 de olan 3. örnek gibi, fiziksel araç – işlevsel kullanıcı – ilgi nesnesinin oluşturduğu veya aldığı veri hareketlerinden ayrıştırılamaz. Bu gibi durumlarda ilgi nesnesini işlevsel kullanıcıdan yarı imiş gibi dokümante etmek fazla değer katmaz. Önemli nokta bu kavramları faydalı oldukları yerlerde ayrı veri gruplarını dolayısıyla ayrı veri hareketlerini ayrıştırmak için kullanmaktır.

ÖRNEK: Varsayalım bir sıcaklık algılayıcısı 'A', bir işlevsel süreç tarafından sorgulandığında anlık sıcaklığı sürece yollamaktadır. İşlevsel kullanıcı algılayıcı A'dır; Giriş mesajı ismi 'A'daki anlık sıcaklık' olabilir; bu mesajın ilgi nesnesi ise yine 'algılayıcı A' olarak değerlendirilebilir. Teorik olarak ilgi nesnesi 'algılayıcı A' değil, 'algılayıcı A tarafından sıcaklığı ölçülen madde veya nesne'dir. Fakat pratikte bu ince ayrımı yapmak dokümana çok az değer katmaktadır ve bu ilgi nesnelerini ayrı tanımlamaya değmeyebilir.

3.4 Veri özniteliklerini belirleme (Tercihe bağlı)

Bu kısım ölçülecek yazılım parçası tarafından referans verilen veri özniteliklerinin tanımlanmasını inceler. Ölçüm yönteminin bu sürümünde, veri özniteliklerini tanımlamak zorunlu değildir. Buna rağmen, veri gruplarının ve ilgi nesnelerinin ayrılması sürecinde veri özniteliklerini analiz etmek ve tanımlamak faydalı olabilir. Veri öznitelikleri, eğer boyut ölçüsünün bir alt-birimi gerekiyorsa, kısım 4.5 'COSMIC Ölçme Yönteminin Genişletilmesi' içerisinde sunulduğu gibi tanımlanabilir.

3.4.1 Tanım

TANIM – Veri özniteliği
Bir veri özniteliği belirlenen bir veri grubunda, yazılımın İşlevsel Kullanıcı Gereksinimleri bakış açısından anlam taşıyan en küçük bilgi paketidir.

ÖRNEK 1: İş uygulamaları alanı bağlamında veri öznitelikleri:

- Veri sözlüğünde kayıtlı olan veri elemanları,
- Kavramsal veya mantıksal veri modelinde yer alan veri öznitelikleri.

ÖRNEK 2: Gerçek zamanlı uygulama yazılımları bağlamında veri öznitelikleri:

- Bir algılayıcıdan alınan sinyalin veri öznitelikleri

- Bir iletimde yer alan mesajın veri öznitelikleri.

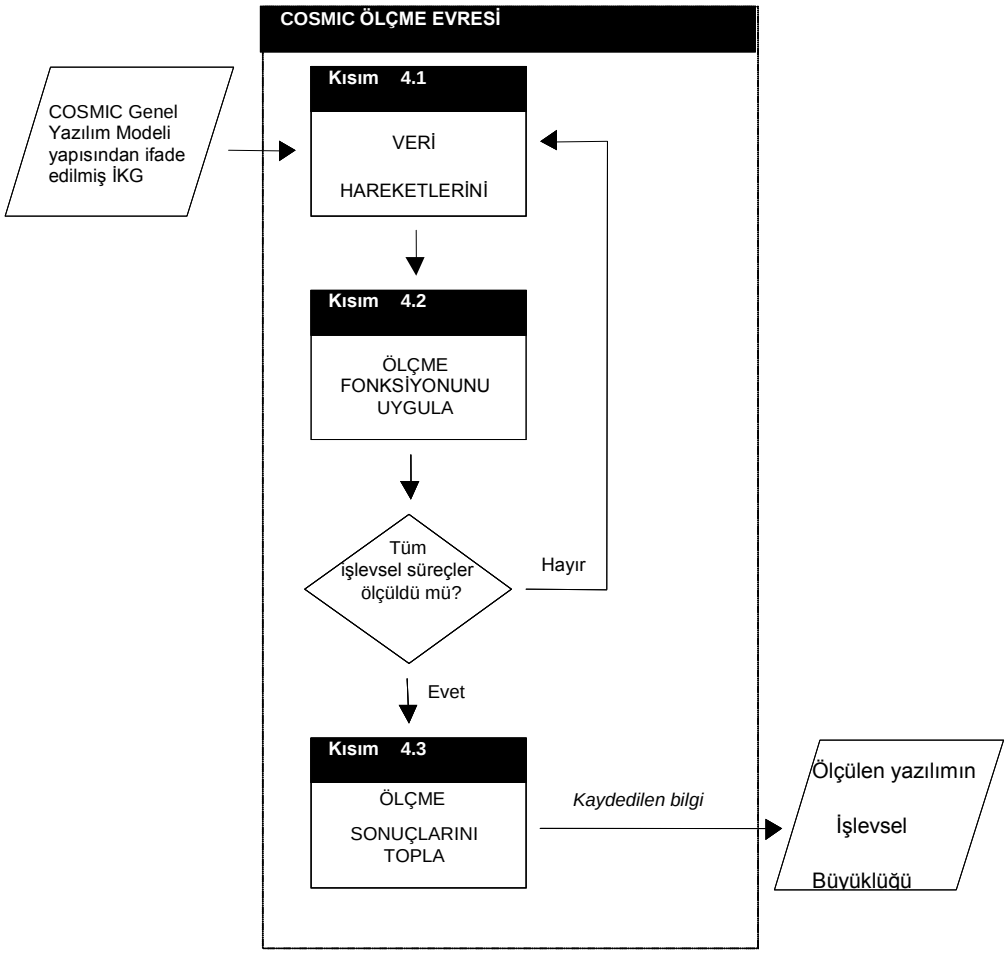
3.4.2 Veri özniteliklerinin ve veri gruplarının ilişkisi hakkında

Teorik olarak, bir veri grubu, eğer yeterliyse, bir ilgi nesnesini anlatmak için İşlevsel kullanıcı Gereksinimleri bakış açısından tek bir veri özniteliği içerebilir. Pratik olarak, bu gibi durumlar çoğunlukla gerçek zamanlı yazılımlarda olurlar (örneğin, Giriş'in gerçek zamanlı saatin bir vuruşunu taşıması gibi); iş uygulamaları yazılımlarında daha az rastlanırlar.

4

ÖLÇME EVRESİ

Bu bölüm COSMIC ölçme sürecindeki ölçme evresine ilişkin kuralları ve yöntemi sunmaktadır. Bir yazılım parçasının İşlevsel Kullanıcı Gereksinimleri, COSMIC Genel Yazılım Modeli olarak ifade edildiğinde nasıl ölçüleceği aşağıdaki şekil 4.0'da özetlenmektedir.



Şekil 4.0 – COSMIC Ölçme Evresi için Genel Süreç

Bu yöntemdeki her basamak, bu bölümde tanımların ve ilkelerin bazı kurallarla ve örneklerle sunulduğu her bir özel alt başlıkta anlatılmaktadır.

4.1 Veri hareketlerinin belirlenmesi

Bu basamak, her işlevsel sürecin veri hareketlerinin (Giriş, Çıkış, Okuma ve Yazma) belirlenmesini içerir.

4.1.1 Veri hareketi tiplerinin tanımı

TANIM – Veri hareketi
Tek bir veri grubu tipini hareket ettiren temel işlevsel bileşen. NOT 1: Veri hareketi tipinin dört çeşit alt tipi vardır: Giriş, Çıkış, Okuma ve Yazma (-tipleri) NOT 2: Ölçme kapsamında, her veri hareketi alt tipinin ilişkili bazı veri işlemelerini de içerdiği varsayılmaktadır - detaylar için bakınız kısım 4.1.6. NOT 3: Daha kesin olarak, gerçekte veri grubu *oluşumlarını* (tiplerini değil) *hareket ettiren* bir veri hareketinin oluşumudur; veri hareketi tipi değildir. Bu açıklama Giriş, Çıkış, Okuma ve Yazma tanımları için de geçerlidir.

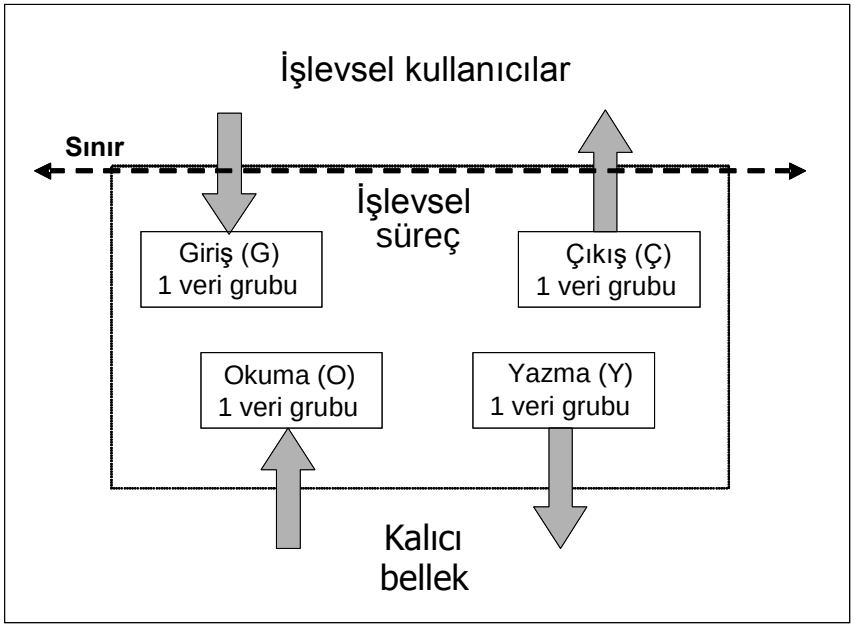
TANIM – Giriş (G)
Bir Giriş (G), bir veri grubunu bir işlevsel kullanıcıdan uygulama sınırından içeriye ihtiyaç duyulan bir işlevsel sürece doğru hareket ettiren veri hareketidir. NOT: Bir Girişin belirli ölçüde ilişkili veri işleme içerdiği kabul edilir (bakınız kısım 4.1.6).

TANIM – Çıkış (Ç)
Bir Çıkış, bir veri grubunu bir işlevsel süreçten uygulama sınırından dışarıya ihtiyaç duyan bir işlevsel kullanıcıya doğru hareket ettiren veri hareketidir. NOT: Bir Çıkışın belirli ölçüde ilişkili veri işleme içerdiği kabul edilmektedir (bakınız kısım 4.1.6).

TANIM – Okuma (O)
Bir veri grubunu kalıcı bellekten ihtiyaç duyulan işlevsel sürece doğru hareket ettiren bir veri hareketidir. NOT: Bir Okumanın belirli ölçüde ilişkili veri işleme içerdiği kabul edilir (bakınız kısım 4.1.6).

TANIM – Yazma (Y)
Bir işlevsel süreçte yer alan veri grubunu kalıcı belleğe doğru hareket ettiren veri hareketidir. NOT: Bir Yazmanın belirli ölçüde ilişkili veri işleme içerdiği kabul edilir (bakınız kısım 4.1.6).

Aşağıda verilen Şekil 4.1.1, dört veri hareketi tipi arasındaki genel ilişkiyi, ait oldukları işlevsel süreci ve ölçülen yazılımın sınırını göstermektedir.



Şekil 4.1.1 – Dört veri hareketi tipi ve bunların işlevsel süreç ve veri grupları ile ilişkileri

4.1.2 Girişlerin (G) belirlenmesi

Belirlendiğinde, bir aday Giriş veri hareketi aşağıdaki ilkelere uygun olmalıdır:

İLKELER – Giriş (G)
a) Bir Giriş tek bir ilgi nesnesini tanımlayan bir veri hareketini işlevsel kullanıcıdan uygulama sınırından kendisinin de bir parçası olduğu işlevsel sürece doğru hareket ettirmelidir. Bir işlevsel sürecin girdisi birden çok veri grubu içeriyor ise, girdideki her bir benzersiz veri grubu için bir Giriş tanımlanır (ayrıca bakınız kısım 4.1.7 - 'Veri Hareketi Benzersizliği').
b) Bir Giriş uygulama sınırından bir veriyi çıkaramaz, veriyi okuyamaz veya yazamaz.
c) Bir işlevsel sürecin işlevsel kullanıcıdan veri alması gerektiği, fakat kullanıcıya hangi verinin gönderileceğinin söylenmesinin gerekli olmadığı veya işlevsel kullanıcının gelen herhangi bir mesaja tepki veremeyeceği durumda, verinin alınması için işlevsel süreçte bir Giriş belirlenir. Bu durumlarda, işlevsel süreçten işlevsel kullanıcıya verinin elde edilmesi için gönderilen herhangi bir mesaj bir Çıkış olarak sayılmamalıdır. Ancak, işlevsel sürecin işlevsel kullanıcıdan veri alması gerektiği ve işlevsel sürecin talebi karşılamak amacıyla işlevsel kullanıcıya veri temin etmesi gerektiği durumda, talep için bir Çıkış ve talep edilen verinin karşılığı olarak bir Giriş belirlenir (daha fazla bilgi için bakınız kısım 4.1.9).

Aşağıdaki kurallar aday bir Giriş veri hareketinin statüsünü onaylamakta yardımcı olacaktır:

KURALLAR – Giriş (G)
a) Tetikleyici bir Giriş'teki veri grubu, basitçe yazılıma 'bir Y olayı oldu' bilgisini veren yalnızca bir veri özniteliğinden oluşabilir. Sıklıkla, özellikle iş uygulamaları yazılımlarında, tetikleyici Girişin veri grubu, yazılıma 'Y olayı gerçekleşti ve bu belirli olay ile ilgili veriler şunlardır' bilgisini veren birçok veri özniteliğini içerir.
b) Tetikleyici olay olan saat-vuruşları her zaman ölçülecek yazılımın dışındadır. Bu

nedenle, örneğin her 3 saniyede meydana gelen bir saat tıklaması, tek bir veri özniteliği olan bir veri grubunu hareket ettiren bir Giriş ile ilişkilendirilmelidir. Tetikleyici olayın belirli aralıklarla donanım veya ölçülen yazılımın sınırları dışında bir donanım veya başka bir yazılım parçası tarafından oluşturulmuş olması bir fark yaratmayacaktır.

c) Sistem saatinden zamanın edinilmesi, ayrı bir işlevsel süreç gerektirmediği sürece, bir Girişe sebep olduğu düşünülmemelidir.

d) Belirli bir olayın oluşu, belirli bir ilgi nesnesinin en çok 'n' sayıda veri özniteliğini içeren bir veri grubunu tetikliyor ise ve İKG, aynı olayın başka oluşlarının, ilgi nesnesinin 'n' sayıdaki özniteliklerinin sadece alt-kümesine ait öznitelikler için değer içeren veri gruplarının Girişlerini tetiklemesine izin veriyorsa, bu durumda tüm 'n' veri özniteliklerini içeren tek bir Giriş tanımlanmalıdır.

Kural c)'yi gösteren bir örnek şu şekilde olabilir: Bir işlevsel süreç zaman damgasını yazıyorsa, sistem saatinin alınması için herhangi bir Giriş belirlenmez.

Belirlendiğinde, her Giriş veri hareketi, Genel Yazılım Modeli matrisinde (Ek A) ilgili hücreye 'G' ile işaretlenerek kaydedilebilir.

4.1.3 Çıkışların (Ç) Belirlenmesi

Belirlendiğinde, bir aday Çıkış veri hareketi aşağıdaki ilkelere uygun olmalıdır:

İLKELER – Çıkış (Ç)

a) Bir Çıkış, tek bir ilgi nesnesini tanımlayan bir veri hareketini kendisinin de bir parçası olduğu işlevsel süreçten uygulama sınırından işlevsel kullanıcıya doğru hareket ettirir. Eğer bir işlevsel süreçten çıktılar birden fazla veri grubundan oluşuyorsa, çıktıdaki benzersiz her bir veri grubu için bir Çıkış belirlenir (ayrıca bakınız kısım 4.1.7 - 'Veri Hareketi Benzersizliği').

b) Bir Çıkış, uygulama sınırından bir veriyi geçiremez, bir veriyi okuyamaz veya yazamaz.

Aşağıdaki kurallar aday bir Çıkış veri hareketinin statüsünü onaylamakta yardımcı olacaktır:

KURALLAR – Çıkış (Ç)

a) Kullanıcı verisi olmadan yazılım tarafından oluşturulan her türlü mesaj ve çıktı (örneğin hata mesajları) tek bir ilgi nesnesine ilişkin tek bir özniteliğe ait değerler olarak kabul edilmelidir ('hata bildirimi' olarak isimlendirilebilir). Bunun için, İKG'de ihtiyaç duyulan ve her işlevsel süreçte bulunan bütün bu mesaj oluşumları tek bir Çıkış olarak belirlenir.

b) Bir işlevsel süreçteki bir Çıkış, bir ilgi nesnesine ilişkin 'n' tane veri özniteliğinden oluşan bir veri grubunu hareket ettiriyorsa ve İKG, işlevsel sürecin bu ilgi nesnesine ilişkin 'n' özniteliğinin bir alt kümesinden oluşan bir veri grubunu hareket ettiren bir Çıkışın oluşumuna müsaade ediyorsa; bu durumda 'n' tane veri özniteliğinden oluşan tek bir Çıkış belirlenmelidir.

Yukarıda verilen a) kuralı için örnekler şu şekildedir:

ÖRNEK 1: Bir insan-bilgisayar diyalogunda, girilen verinin geçerlenmesi sırasında oluşan hata mesajı örnekleri 'biçim hatası', 'müşteri bulunamadı', 'hata: şartlar ve koşullar okunmuştur onay kutucuğunu işaretleyiniz', 'kredi limiti aşıldı', vb. şeklinde olabilir. Tüm bu hata mesajları her bir işlevsel süreç içinde, bu mesajların oluştuğu durumlarda, tek bir Çıkış oluşu ('hata mesajları olarak adlandırılabilir') olarak değerlendirilmelidir.

ÖRNEK 2: Bir uygulamanın ölçümünde, insan kullanıcılarına çıktı olan fakat uygulama yazılımı tarafından oluşturulmamış hata mesajları tamamıyla göz ardı edilmelidir. Bu tür bir mesaja örnek olarak işletim sistemi tarafından oluşturulan 'yazıcı X cevap vermiyor' hata mesajı verilebilir.

ÖRNEK 3: Bir insan-bilgisayar diyalogunda, bir mesaj hata durumlarında oluşturuluyor fakat işlevsel kullanıcı verisi içeriyorsa, bu mesajların oluştuğu durumlarda, mesaj bir Çıkış olarak sayılmalıdır. Bu tip bir mesaja örnek olarak 'Uyarı: çekmek istediğiniz miktar, çekme limitinizin 100 Dolar üzerindedir (100 Dolar hesaplanan bir değişken olmak üzere)' verilebilir. Bu örnekte Çıkış, müşterinin banka hesabı ile ilgili veri grubu içermektedir.

ÖRNEK 4: Bir gerçek-zamanlı sistemde, tüm donanım aygıtlarının doğru çalıştığını belirli aralıklarla denetleyen bir işlevsel süreç, 'X' bir değişken olmak üzere, 'Sensor X başarısız oldu' bilgisini raporlayan bir hata mesajı verebilir. Bu mesaj o işlevsel süreçte tek bir Çıkış olarak tanımlanmalıdır ('X' değişkeninin değerinden bağımsız olarak).

ÖRNEK 5: A ve B iki işlevsel süreç olsun. 'A', işlevsel kullanıcılarına potansiyel olarak 2 farklı onay mesajı ve 5 hata mesajı verebilir ve 'B', işlevsel kullanıcılarına potansiyel olarak 8 hata mesajı verebilir. Bu örnekte 'A' işlevsel süreci için (5+2=7 mesaja istinaden) tek bir Çıkış belirlenmeli ve 'B' işlevsel süreci için (8 mesaja istinaden) ayrı bir Çıkış belirlenmelidir.

Belirlendiğinde, her Çıkış veri hareketi, Genel Yazılım Modeli matrisinde (Ek A) ilgili hücreye 'Ç' ile işaretlenerek kaydedilebilir.

4.1.4 Okumaların (O) Belirlenmesi

Belirlendiğinde, bir aday Okuma veri hareketi aşağıdaki ilkelere uygun olmalıdır:

İLKELER – Okuma (O)
a) Bir Okuma, tek bir ilgi nesnesini anlatan tek bir veri grubunu kalıcı bellekten, Okumanın bir parçasını oluşturduğu işlevsel sürece hareket ettirmelidir. Bir işlevsel süreç kalıcı bellekten birden çok veri grubu getirmek durumunda ise, getirilen her bir benzersiz veri grubu için bir Okuma tanımlanır (ayrıca bakınız kısım 4.1.7 - 'Veri Hareketi Benzersizliği').
b) Bir Okuma, sınır dışına veri çıkarmamalı, sınır dışından veri almamalı veya veri yazmamalıdır.
c) Bir işlevsel süreç sırasında, işlevsel sürecin dahilinde gerçekleşen ve sadece programcı tarafından değiştirilebilir sabit veya değişkenlerin hareket ettirilmesi veya işletilmesi; veya İKG'nden ziyade sadece uygulamadan kaynaklanan, ara sonuçların veya işlevsel sürecin depoladığı verilerin hesaplanması, Okuma veri hareketi olarak değerlendirilmemelidir.
d) Bir Okuma veri hareketi her zaman 'Okuma isteği' işlevini içerir (bu yüzden, 'okuma isteği' işlevi hiçbir zaman ayrı bir veri hareketi olarak sayılmamalıdır). Ayrıca bakınız kısım 4.1.9.

Belirlendiğinde, her Okuma veri hareketi, Genel Yazılım Modeli matrisinde (Ek A) ilgili hücreye 'O' ile işaretlenerek kaydedilebilir.

4.1.5 Yazmaların (Y) Belirlenmesi

Belirlendiğinde, bir aday Yazma veri hareketi şu ilkelerine uyumlu olmak zorundadır:

İLKELER – Yazma (Y)
a) Bir Yazma, tek bir ilgi nesnesini anlatan tek bir veri grubunu, Yazmanın bir parçasını oluşturduğu işlevsel süreçten, kalıcı belleğe hareket ettirmelidir. Bir işlevsel süreç, kalıcı belleğe birden çok veri grubu hareket ettirmek durumunda ise, kalıcı belleğe hareket ettirilen her bir benzersiz veri grubu için bir Yazma tanımlanır (ayrıca bakınız kısım 4.1.7 - 'Veri Hareketi Benzersizliği').
b) Bir Yazma, sınır dışından veri almamalı, sınır dışına veri çıkarmamalı veya veri okumamalıdır.
c) Kalıcı bellekten bir veri grubunun silinmesine ilişkin bir gereksinim, tek bir Yazma veri hareketi olarak ölçülmelidir.
d) Bir işlevsel süreç sırasında, işlevsel süreç tamamlandığında kalıcı olmayan verinin hareket ettirilmesi veya işletilmesi; işlevsel sürecin dahilinde olan değişkenlerin

güncellenmesi veya hesaplamalarda ara sonuçların üretilmesi Yazma veri hareketi olarak değerlendirilmemelidir.

Belirlendiğinde, her Yazma veri hareketi, Genel Yazılım Modeli matrisinde (Ek A) ilgili hücreye 'Y' ile işaretlenerek kaydedilebilir.

4.1.6 Veri hareketleri ile ilişkili veri işlemeleri üzerine

Genel Yazılım Modelinin (d) ilkesinde (bakınız kısım 1.4) tanımlandığı üzere, alt-süreçler veri hareketleri veya veri işlemeleridir. Ancak, mevcut COSMIC kuralları gereği (bakınız Genel Yazılım Modeli ilke (j)), veri işleme alt-süreçlerinin ayrı şekilde varlıkları tanınmamıştır.

TANIM – Veri işleme
İşlevsel süreç dahili ve haricindeki veya işlevsel süreç ile kalıcı bellek arasındaki veri hareketleri dışında veride meydana gelen her türlü şey.

Aşağıda verilen ilke COSMIC yönteminin veri işlemelerini nasıl ele aldığını belirler.

İLKE – Veri Hareketleri ile ilişkili veri işleme
Bir işlevsel süreçteki tüm veri işlemeleri dört veri hareketi tipi (G,Ç,O,Y) ile ilişkilendirilmelidir. Kural gereği, bir işlevsel sürecin veri hareketlerinin, işlevsel sürecin veri işlemelerini de temsil ettiği varsayılır.

Hangi veri işleme tiplerinin hangi veri hareketi tipleri ile ilişkili olduklarını tanımlama ihtiyacı, sadece yazılımdaki *değişikliklerin* ölçülmesi (bakınız kısım 4.4) sırasında ortaya çıkar. Gereken tipik bir değişiklik, hem hareket ettirilen öznitelikleri hem de belirli bir veri hareketi ile ilişkili veri işlemesini etkiler, fakat verinin hareketini değil sadece verinin işlemesini *etkileyebilir*. Buna rağmen, bu tip bir değişikliğin belirlenmesi ve ölçülmesi gereklidir. Buna göre, bir işlevsel süreç içindeki veri işlemesini değiştirecek herhangi bir gereksinim olduğunda, ölçücünün veri işlemesine yapılan değişikliğin hangi veri hareketi ile ilişkili olduğunun belirlenmesi gerekir.

Veri işlemesinin belirlenmesi için genel ilkeler, her bir veri hareketi için aşağıda sunulmuştur.

Giriş veri hareketi

Bir Giriş aşağıdakilerle ilgili tüm veri işlemelerini içerir:

- Bir veri grubunun işlevsel kullanıcı tarafından girilmesinin sağlanması (ör. formatlama ve sunum işlemeleri) ve/veya doğrulanması.
- Fakat başka bir veri hareketi içeren veya veri grubunun girilmesi ve doğrulanmasından sonraki işlemeleri içermez.

ÖRNEK: Bir Giriş, girilen kod/verilerin doğrulanması veya ilgili tanımların elde edilmesi için gereken Okuma(lar) HARİÇ girilen verinin ekranındaki tüm veri işleme, formatlama ve sunumları içerir.

İşlevsel kullanıcıya nasıl bir veriye gerek duyulduğunun bildirilmesi gerektiği durumlar hariç bir Giriş veri hareketi 'giriş isteği' işlevini içermelidir (nasıl bir veri gönderilmeli ile ilgili bakınız kısım 4.1.9 ve boş girdi ekranlarının değerlendirilmesi ile ilgili bakınız kısım 4.1.10).

Çıkış veri hareketi

Bir Çıkış aşağıdakilerle ilgili tüm veri işlemelerini içerir:

- Çıktı olacak bir veri grubuna ait veri öznitelikleri nin oluşturulması ve/veya
- Veri grubunun istenilen işlevsel kullanıcıya çıktı olmasının (ör. formatlama ve sunum işlemeleri) ve ulaştırılmasının sağlanması.
- Fakat başka bir veri hareketi içeren işlemeleri içermez.

ÖRNEK: Bir Çıkış, basılacak bazı veri özniteliklerinin değerleri veya ilgili tanımların sağlanması için gerekebilecek Okuma(lar) veya Girişler HARİÇ insan-kullanıcı tarafından okunabilir alan başlıkları[13] dahil olmak üzere, veri özniteliklerinin formatlanması ve basılmaya hazırlanması ile ilgili tüm işlemleri içerir.

Okuma veri hareketi

Bir Okuma, bir veri grubunun kalıcı bellekten alınması için gereken tüm işlemleri ve/veya hesaplamaları içerir; fakat başka bir veri hareketi içeren veya Okumanın başarılı şekilde bitirilmesinin ardından yapılan işlemeleri içermez.

ÖRNEK: Bir Okuma, bir veri grubunun kalıcı bellekten alınması için gereken tüm matematiksel hesaplamaları ve mantıksal işlemleri içerir; fakat veri grubunun alınmasının ardından yapılan veri öznitelikleri işlemelerini içermez.

Bir Okuma, herhangi bir 'Okuma isteği' işlevini de her zaman içerir (bakınız kısım 4.1.9).

Yazma veri hareketi

Bir Yazma, yazılacak bir veri grubunun yaratılması için gereken tüm işlemleri ve/veya hesaplamaları içerir; fakat başka bir veri hareketi içeren veya Yazmanın başarılı şekilde bitirilmesinin ardından yapılan işlemeleri içermez.

ÖRNEK: Bir Yazma, yazılacak veya silinecek veri grubu içindeki herhangi bir veri özniteliğinin değerinin sağlanması için gerekebilecek Okumalar veya Girişler HARİÇ, yazılacak veya silinecek bir veri grubunun yaratılması veya güncellenmesi için gereken tüm matematiksel hesaplamaları ve mantıksal işlemleri içerir.

4.1.7 Veri hareketi tekilliği ve olası istisnalar[14]

Genel Yazılım Modeli, normal olarak herhangi bir işlevsel süreçte herhangi bir ilgi nesnesini tanımlayan tüm verinin bir Giriş veri hareketi tipi ile girildiğini ve/veya bir Okuma veri hareketi tipi ile okunduğunu ve/veya bir Çıkış veri hareketi tipi ile çıktı olduğunu ve/veya bir Yazma veri hareketi tipi ile yazdırıldığını varsayar. Model ayrıca, bir veri grubuna ait veri özniteliklerinin olası tüm değerlerinin hareket ettirilmesinden kaynaklanan tüm veri işlemelerinin tek bir veri hareketi ile ilişkili olduğunu varsayar.

Son kısımda verilen varsayımı örneklemek için, bir işlevsel sürecin (- süreç tipinin) iki ayrı oluşumunu değerlendirelim. Varsayalım ki; birinci oluşumda hareket ettirilen bazı öznitelik değerlerinin veri işleme alt-süreç (tipi) A'ya neden olurken, ve aynı işlevsel sürecin bir diğer oluşumunda öznitelik değerleri farklı bir veri işleme alt-süreç (tipi) B'ye neden oluyor. Bu şartlar altında, her iki veri işleme alt-sürecinin – 'A' ve 'B'-, aynı ve tek bir veri hareketi ile ilişkilendirilmesi ve böylece o işlevsel süreçte sadece bir veri hareketinin belirlenmesi ve sayılması gerekir.

Bununla birlikte, bir ilgi nesnesini tanımlayan aynı veri grubu tiplerinin, aynı tipte (G, Ç, O, Y) ve aynı işlevsel süreç içindeki bir veri hareketinde hareket ettirilmesinin istendiği (İKG'nde) istisnai durumlar olabilir. Alternatif ve yine istisnai bir durum olarak, aynı veri hareketi tipi (G, Ç, O, Y) ve aynı işlevsel süreç içinde, fakat farklı veri işlemeleri ile ilişkili olan aynı veri grubunun hareket ettirilmesi istenmiş olabilir.

Aşağıda verilen kural ve örnekler, olağan durumları, olası geçerli istisnaları ve geçerli gibi gözüküp fakat geçeriz olan örnekleri kapsamaktadır.

KURAL – Veri hareketi benzersizliği ve olası istisnalar
a) İşlevsel Kullanıcı Gereksinimleri aksini belirtmediği sürece, bir işlevsel sürece giriş olması gereken herhangi bir ilgi nesnesini tanımlayan tüm veri öznitelikleri ve ilişkili tüm veri işlemeleri tek bir Giriş (tipi) olarak belirlenmeli ve sayılmalıdır.

[13] Bu örnek alandan bağımsız olarak insan kullanımı için geliştirilmiş yazılım uygulamalarının ölçümü için uygulanabilir. Bu, açıktır ki, giriş ve çıkış ekranlarında özel alan başlıklarını destekleyen yeniden-kullanılabilir objelerin ölçülmesinde uygun değildir.

[14] Ölçme Kılavuzunun 2.2 sürümünde, bu kısmın başlığı 'Veri hareketlerinin tekilleştirilmesi (de-dublication)' olarak adlandırılmıştır. 'tekilleştirme (de-dublication)' teriminin yeterince açıklayıcı olmadığı düşünülerek, terminoloji değiştirilmiştir.

(Not: Bir işlevsel süreç, elbette ki, her biri farklı bir ilgi nesnesini (tipini) tanımlayan bir veri grubunu hareket ettiren, birden çok Giriş tipini ele alması/içermesi gerekebilir.

Aynı eşdeğer kural herhangi bir işlevsel süreçteki herhangi bir Okuma, Yazma veya Çıkış veri hareketi için geçerlidir.

b) Birden çok Giriş için bir İşlevsel Kullanıcı Gereksinimi var ise, bir işlevsel süreçte her biri aynı ilgi nesnesini (tipini) tanımlayan veri gruplarını hareket ettiren birden çok Giriş veri hareketi (tipi) belirlenebilir ve sayılabilir. Benzer olarak, birden çok Giriş için bir İşlevsel Kullanıcı Gereksinimi var ise, aynı işlevsel süreçte aynı veri grubunu (tipini) hareket ettiren fakat her biri farklı veri işlemesi (tipi) ile ilişkili birden çok Giriş veri hareketi (tipi) belirlenebilir ve sayılabilir.

Bu tür İKG, birden çok Girişin bir işlevsel süreçte (aynı ilgi nesnesini tanımlayan) farklı veri gruplarını giren farklı işlevsel kullanıcıların varlığından kaynaklandığı durumlarda ortaya çıkabilir.

Aynı eşdeğer kural herhangi bir işlevsel süreçteki herhangi bir Okuma, Yazma veya Çıkış veri hareketi için geçerlidir.

c) Tekrarlanan veri hareketi tipi *oluşumları* (yani aynı veri işlemelerine sahip aynı veri grubunun hareketi) herhangi bir işlevsel süreçte birden çok kez belirlenmemeli ve sayılmamalıdır.

d) Eğer, bir işlevsel süreçteki veri hareketinin çoklu *oluşumu*, ilgili veri işlemeleri bazında, veri grubuna ait farklı veri öznitelikleri *değerlerinin* hareket ettirilmesinin farklı işleme yollarına neden olması dolayısıyla değişiyorsa, veri hareketi tipi o işlevsel süreçte birden çok kez belirlenmemeli ve sayılmamalıdır.

Yukarıdaki kurallar aşağıda verilen örnekler ile açıklamaktadır.

ÖRNEK 1 (kural 'a)' için): Herhangi bir işlevsel süreç içinde, belirli bir ilgi nesnesini tanımlayan verinin tüm Okumaları mantıksal olarak o ilgi nesnesini tanımlayan tüm öznitelikleri geri döndürdüğü düşünülebilir. Dolayısıyla, herhangi bir işlevsel süreçte genel olarak herhangi bir ilgili nesnesi ile ilgili herhangi bir verinin sadece bir Okumasının olması işlevsel olarak gereklidir ve bu şekilde belirlenmelidir.

ÖRNEK 2 (kural 'a)' ve 'b)' için): Örnek 1 takip edilecek olursa, okunan her veri bir Yazma komutu ile kalıcı hale getirilmiş olduğundan, Kural 'a)' gereği, bir işlevsel süreç içinde kalıcı hale getirilmesi gerekli bir ilgi nesnesinin tüm veri özniteliklerini içeren veri grubunu hareket ettiren tek bir Yazma olması genel bir durumdur. Ancak, istisnai olarak aynı ilgi nesnesini tanımlayan iki ayrı veri grubunun yazıldığı tek bir işlevsel süreç için bir İKG tanımlanmış olabilir; örneğin, başka işlevsel süreçlerde iki ayrı işlevsel kullanıcının daha sonraki kullanımları için. Kural 'b)'nin geçerli olacağı şu şekilde bir örnek verilebilir: Tek bir işlevsel sürecin (A) bir bankanın mevcut hesap dosyalarından daha sonra iki ayrı program (uygulama) için kullanılacak iki alt-küme veri çıkarması gerekebilir. İlk alt-küme 'hesap mevcudundan daha fazla çekilen hesap detaylarından' (eksi bakiye özniteliğini içeren) oluşur. İkinci alt-küme 'yüksek değerli hesap detaylarından' (pazarlama mesajları için hesap sahibinin adı ve adresini içeren) oluşur. İşlevsel süreç A her iki alt-küme için iki Yazma içerecektir.

ÖRNEK 3 (kural 'b)' için: Aynı ilgi nesnesini tanımlayan farklı veri gruplarını hareket ettiren ve farklı işlevsel kullanıcılara yönelik olarak iki yada daha fazla Çıkışa neden olan tek bir işlevsel süreç için bir İKG tanımlanmış olabilir. Örneğin, yeni bir çalışanın firmaya katılımında, verinin geçerliliğinin çalışan tarafından teyidi için bir rapor hazırlanır ve binaya giriş yetkisi için Güvenliğe mesaj gönderilir.

ÖRNEK 4 (kural 'c)' için: Bir İKG'da, pratikte birçok bulup getirme içeren oluşumların gerekli olduğu (bir doküman içinde yapılan arama örneğinde olduğu gibi) bir Okumanın gerektiğini varsayalım. Ölçme amacı doğrultusunda, tek bir Okuma belirleyiniz.

ÖRNEK 5 (kural 'c)' için: Gerçek zamanlı bir işlevsel süreçte, bir İKG bir verinin bir işlevsel kullanıcı tarafından girilmesini (örneğin, bir donanım aygıtının değişim hızının ölçümü veya süreç içinde bir değerin değişip değişmediğinin kontrol edilmesi için belirli bir zaman aralığında iki kere) gerektirebilir. Hareket ettirilen veri grupları ve ilişkili veri işlemeleri açısından iki Girişin özdeş olması şartı ile, sadece tek bir Giriş belirlenmelidir. (Bir Giriş ile ilişkili olduğu değerlendirilen veri işleme tipleri için bakınız kısım 4.1.6.)

ÖRNEK 6 (kural 'c)' için: Girişler için bakınız kısım 4.1.2, Kural d) ve Çıkışlar için bakınız kısım 4.1.3, Kural b).

ÖRNEK 7 (kural 'd)' için: Bir Girişin veri öznitelikleri değerlerine bağlı olarak çeşitli veri işleme seçeneği sunan bir işlevsel sürecin gerektiğini varsayalım. Ölçme amacı doğrultusunda, tek bir Giriş belirleyiniz.

ÖRNEK 8: İKG'nde bir veri grubu Okumasının gerektiğini varsayalım; fakat geliştirici, işlevsel süreç içinde değişik noktalarda aynı ilgi nesnesine ait veri özniteliklerinin farklı alt-kümelerinin kalıcı bellekten iki ayrı komut ile okunmasına karar vermiş olsun. Bu durumda tek bir Okuma belirleyiniz.

4.1.8 Bir işlevsel süreç kalıcı belleğe veya kalıcı bellekten veri hareket ettirdiğinde

Bu kısım, bir uygulama yazılımı işlevsel sürecinin lokal veya uzak bir kalıcı belleğe veya o bellekten veri hareket ettirmesi ile ilgili veri hareketlerini açıklamaktadır. Bu örnek olaylar ayrıca, uygulamanın bellek ihtiyaçlarının kalıcı bellek sürücüsü gibi yazılımı başka katmanlarda destekleyen diğer yazılımlar tarafından nasıl karşılandığını da göstermektedir.

Örnekler, Yazılım Bağlam Modeli (g) ilkesinin ve Genel Yazılım Modeli ilkelerinin uygulamasını göstermektedir. Örneklerin anlaşılmasında anahtar nokta, bu ilkelerin ölçülmesi gereken her yazılım parçası için ayrı ayrı uygulanma zorunluluğudur.

İlk örnek, bir yazılımda kalıcı verinin getirilmesi gereksiniminin başka bir katmandaki lokal cihaz-sürücüsü yazılımı tarafından ele alındığı sorgu veri hareketleri ile ilgilidir. İkinci örnek olay, veri getirme gereksinimini, öncelikle uygulama ile aynı katmandaki eşdüzey bir yazılım parçası ile karşılandığı durumlarda veri hareketlerinin nasıl değiştiğini göstermektedir.

Pratiklik açısından örnekler, iki yazılım parçasının hiyerarşik bir ilişkiye (her iki yazılım ayrı katmanlarda) veya istemci-sunucu ilişkisine (her iki yazılım aynı katmanda) sahip olduğu durumlarda bu iki yazılımın ölçümü için uygundur. Örnekler, ölçülecek iki yazılım parçası arasında fiziksel olarak gidip-gelen veri hareketlerinin nasıl modellendiğini göstermektedir.

Örnekler Mesaj Sıralama Diyagramı gösterim kuralları kullanılarak gösterilmiştir. Bu tip diyagramların notasyonu şu şekildedir:

- Aşağıya yönelmiş kalın dikey oklar bir işlevsel süreci ifade eder.
- G, Ç, O, ve Y, sırasıyla Giriş, Çıkış, Okuma ve Yazma olmak üzere, yatay oklar veri hareketlerini ifade eder. Giriş ve Okumalar işlevsel sürece giren oklar şeklinde, Çıkış ve Yazmalar çıkan oklar şeklinde gösterilir ve işlevsel süreçte belirtildiği üzere yukarıdan aşağıya sıra ile belirirler.
- Dikey noktalı çizgi bir sınırı ifade eder.

ÖRNEK 1: Bir işlevsel süreç lokal kalıcı belleğe veya lokal kalıcı bellekten veri hareket ettirmesi gerektiğinde

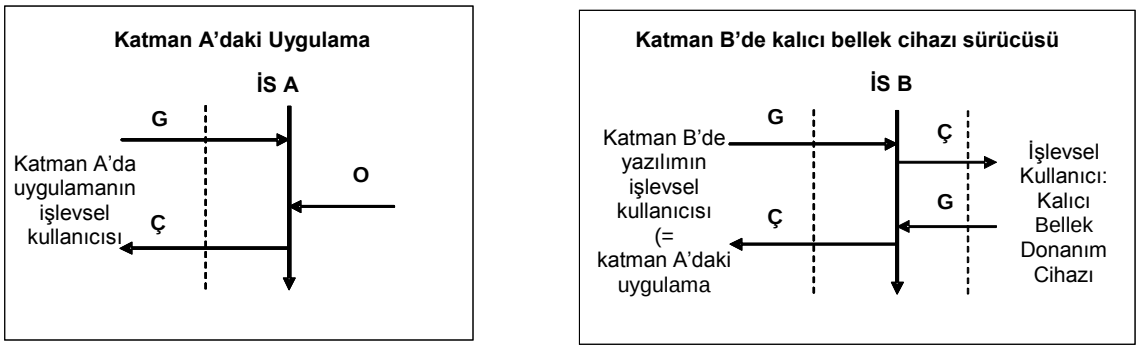
Bu Örnek, uygulama yazılımı 'A' ve bu uygulamanın kullandığı ayrı bir yazılım parçası olan kalıcı bellek cihazı sürücüsü yazılım 'B'den oluşan iki yazılım parçasını kapsamaktadır. (Kolaylık adına olası işletim sistemi varlığını yok sayıyoruz ki; işletim sistemi fiilen uygulama isteklerini cihaz sürücüsü yazılımına iletir ve bu isteklerin sonuçlarını döndürür.)

Katman kavramı bize yazılımların farklı katmanlarda olduklarını ifade eder: uygulama katmanı ve cihaz sürücüsü katmanı. Fiziksel olarak, Şekil 4.1.8.1'deki örnekte de gösterildiği üzere, bu yazılımlar arasında hiyerarşik bir ilişki ve bu iki katmandaki yazılımlar arasında fiziksel bir arayüz vardır (işletim sistemini yok sayarak).

Normal olarak, uygulama 'A'nın İKG'si kalıcı bellekten Okuma veya belleğe Yazmaları içeren işlevsel süreçler tanımlayacaktır, fakat bu Yazma ve Okumaların diğer altyapı yazılımları tarafından nasıl ele alındığı ile ilgilenmeyecektir.

COSMIC modellerinin bu iki yazılıma uygulanmasıyla, uygulama katmanındaki yazılım A'nın işlevsel kullanıcıları örneğin, insan kullanıcıları olabilecekken, sürücü katmanındaki yazılım B'nin işlevsel kullanıcısı (işletim sistemini yok sayarsak) uygulama yazılımı A olacaktır.

Uygulama katmanında yazılım A'ya ait bir sorgu işlevsel süreci 'İS A'nın bir Okuma veri hareketi gerektirdiğini düşünelim. Şekil 4.1.8.1 (a) bu uygulama sorgusunun COSMIC modelini göstermektedir. Kalıcı bellekten gereken verinin fiziksel olarak alınması, sürücü katmanındaki yazılım B'ye ait işlevsel süreç 'İS B' tarafından gerçekleştiriliyor. Şekil 4.1.8.1 (b) cihaz sürücüsüne ait bu işlevsel sürecin modelini göstermektedir.



Şekil 4.1.8.1 (a) ve (b) Uygulama katmanındaki yazılım 'A' tarafından cihaz sürücüsü katmanındaki yazılım 'B'ye bildirilen bir Okuma için Çözüm

Şekil 4.1.8.1 (a), bir Giriş ile tetiklenen, bir Okuma ve sonra sorgu sonucu ile birlikte bir Çıkış ile takip eden sorguyu göstermektedir. İS A'nın verinin nereden edinildiğine veya pratikte Okumanın bir cihaz sürücü yazılımına devredildiğine dair bir bilgisi yoktur.

Şekil 4.1.8.1 (b), işlevsel olarak uygulama A'nın Okuma isteğinin, İS B işlevsel sürecine tetikleyici olay olarak alındığını, daha sonra İS B'nin istenilen veriyi fiziksel kalıcı bellekten bir Giriş/Çıkış çifti ile aldığını, ve veriyi uygulamaya bir Çıkış ile döndürdüğünü göstermektedir. Bu şekilde, uygulama A ve kalıcı bellek donanım aygıtı, cihaz sürücü yazılımı B'nin işlevsel kullanıcılarıdırlar.

Uygulama katmanındaki yazılımın 'Okuma' veri hareketi sayıları ile, cihaz sürücüsü katmanındaki yazılımın 'Giriş/Çıkış çifti' sayıları arasındaki belirgin uyumsuzluk, kural gereği Okuma veri hareketinin herhangi bir 'okuma isteğini' kapsama zorunluluğundan kaynaklanmaktadır.

Tamamen benzer modeller, işlevsel süreç İS A'nın bir veriyi Yazma veri hareketi ile kalıcı hale getirmesi durumunda da uygulanır. Bu örnekte, cihaz sürücü yazılımında İS B işlevsel sürecinin Uygulama A'ya Çıkışı, 'dönüş kodu' veya hata mesajını içerir.

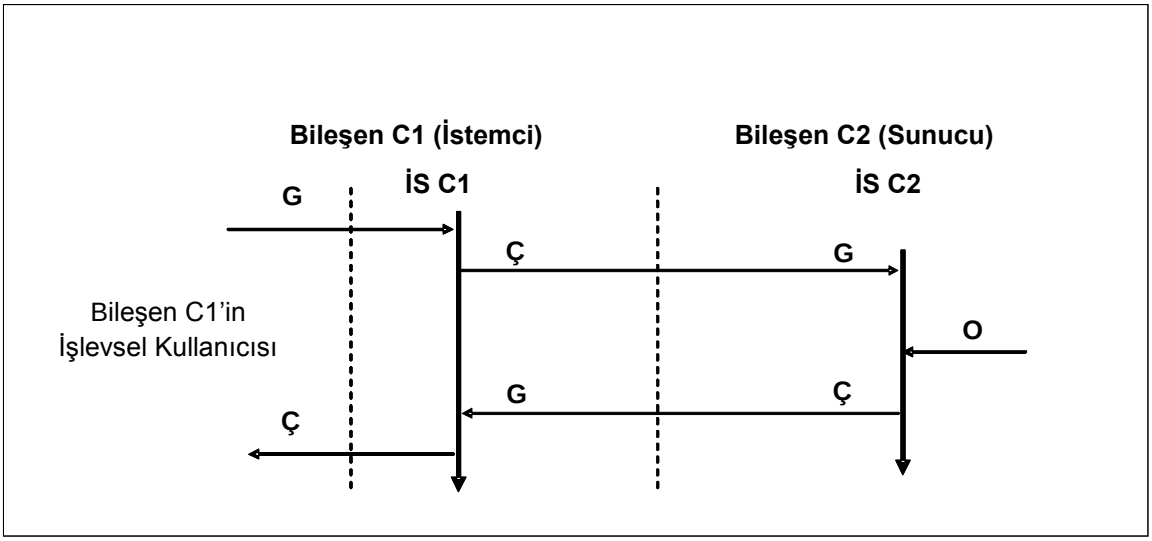
ÖRNEK 2: Bir işlevsel süreç eşdüzey bir yazılımdan veri alması gerektiğinde

Bu örnekte, ölçülecek iki eşdüzey yazılımın 'istemci/sunucu' ilişkisine sahip oldukları, bir başka değişle, istemci yazılımının aynı katmandaki sunucudan servis veya veri aldığı varsayılmaktadır. Şekil 4.1.8.2, aynı uygulamaya ait iki temel (eşdüzey) bileşen arasındaki bu tür bir ilişkinin örneğini göstermektedir. Aynı tür ilişki ve aynı şekil iki yazılımın birinin diğerinden veri alması gerektiği ayrı eşdüzey uygulama olmaları durumunda da geçerli olacaktır.

Fiziksel olarak, bu iki eşdüzey bileşen farklı işlemcilerde yürütülüyor olabilir; bu durumda bileşenler, Şekil 2.2.4.1'de örneği gösterilen bir yazılım mimarisi içinde işlemcilerin ilgili işletim sistemleri ve daha başka ara katmanlar aracılığıyla veri alışverişinde bulunurlar. Fakat mantıksal olarak, COSMIC modellerinin uygulanmasıyla, bileşenler birbirleri ile Giriş/Çıkış çiftleri' aracılığıyla veri alışverişinde bulunurlar. Bu modelde, arada müdahil olan tüm yazılım ve donanımlar göz ardı edilir (Şekil 3.1.1'in sağ kısmında da gösterildiği üzere).

Şekil 2.2.4.1, istemci bileşeni C1'e ait işlevsel süreç İS C1'in işlevsel kullanıcısı tarafından parametreler içeren (örneğin sorgu parametreleri) bir Giriş ile tetiklendiğini göstermektedir. C1 bileşeninin İKG'si, bu bileşenin sunucu bileşeni C2'ye gereken veriyi ve bu verinin ne olduğunu iletmesi gerektiğini onaylayacaktır.

Dolayısıyla, gerekli verinin elde edilmesi için İS C1, C2 bileşenine sorgu isteği parametrelerine sahip bir Çıkış yayınlar. Bileşen C1 bu şekilde bileşen C2'nin işlevsel kullanıcı olmuştur, dolayısıyla bu iki bileşen arasında bir sınır vardır. Bu Çıkış veri hareketi, C1 ve C2 arasındaki sınırı geçer ve böylelikle bileşen C2'ye ait işlevsel süreç İS C2'nin tetikleyici Girişi haline gelir. Bileşen C2'nin işlevsel süreci İS C2, gereken veriyi bir Okuma ile elde eder, bir Çıkış ile veriyi tekrar C1'e gönderir. Bileşen C1'in işlevsel süreci İS C1 veri hareketini bir Giriş olarak alır. İS C1 daha sonra veriyi işlevsel kullanıcı sorgusunun karşılanması için bir Çıkış olarak iletir. Dolayısıyla, Örnek 2'deki bu sorgu, uygulama katmanındaki sorgu isteğini karşılamak için 7 veri hareketine ihtiyaç duyar. Bu, bileşen C1'in veriyi 'lokal' kalıcı bellekten alabiliyor olsaydı (Şekil 4.1.8.1 (a)'da gösterildiği şekilde) uygulama katmanında gerekli olacak 3 veri hareketi (1 x G, 1 x O ve 1 x Ç) ile karşılaştırılabilir.



Şekil 4.1.8.2 Eşdüzey bileşenler arasında veri alışverişi

Elbette ki, Bileşen C2 yüksek ihtimalle, Örnek 1'de de olduğu gibi donanımdan verinin elde edilebilmesi için yazılım mimarisinin daha alt katmanlarındaki herhangi bir kalıcı bellek cihaz sürücüsü servisini kullanacaktır.

Örnek 1 ve 2'yi karşılaştırdığımızda, Örnek 1'de, Örnek 2'deki durumda olduğu gibi uygulama A ile cihaz sürücüsü B'nin modellerinin birleştirilemeyeceğini görüyoruz. Bunun sebebi Okumanın sınırdan geçmemesidir. Şekil 4.1.8.1 (b), uygulama A'nın cihaz sürücüsü yazılımı B'nin işlevsel kullanıcısı olduğunu göstermektedir. Fakat bunun tersi doğru değildir, ki bu, farklı katmanlardaki yazılımların hiyerarşik doğasını gösterir.

Buna karşın, Şekil 4.1.8.2 her iki bileşeni tek modelde gösterebilir, çünkü bu bileşenler aynı katmanda eşdüzey veya eşit bileşenler olarak veri alışverişi yaparlar. Bileşen C1, bileşen C2'nin işlevsel kullanıcısıdır ve tersi de doğrudur, ve ortak bir sınırı paylaşırlar.

Bu iki örnekte kolaylık adına, uygulama A veya bileşen C1 tarafından oluşturulan, Okuma hareketine eşlik eden 'dönüş kodundan' (return code) kaynaklanabilecek hata mesajı Çıkışını (sorgu sonuçlarını içeren Çıkış da dahil olmak üzere) göz ardı ettiğimizi dikkate alınız.

4.1.9 Bir işlevsel süreç işlevsel kullanıcıdan veri temin etmesi gerektiğinde

Bir Giriş için uygulanacak ilke c) gereği (bknz. Kısım 4.1.2), bir işlevsel süreç işlevsel kullanıcıdan veri temin etmesi gerekiyor ise, iki durum söz konusudur. İşlevsel süreç, işlevsel kullanıcıya hangi veriyi göndereceğini bildirmiyor ise, (her ilgi nesnesi için) tek bir Giriş yeterli olacaktır. İşlevsel süreç, işlevsel kullanıcıya hangi veriyi göndereceğini bildirmek durumunda ise, bir Giriş/Çıkış çifti gerekli olacaktır. Aşağıda sunular kurallar uygulanır:

KURALLAR – Bir işlevsel süreç işlevsel kullanıcıdan veri temin etmesi gerektiğinde
a) Aşağıdaki dört durumda olduğu üzere, bir işlevsel süreç işlevsel kullanıcıya hangi veriyi göndereceğini bildirmesi gerekmiyor ise, o işlevsel süreç işlevsel kullanıcıdan bir veri grubunu bir Giriş veri hareketi ile edinmelidir:
• bir işlevsel kullanıcı, işlevsel süreci başlatan bir tetikleyici Giriş gönderdiğinde;
• bir işlevsel süreç, tetikleyici Giriş almasının ardından, duraksayıp işlevsel kullanıcıdan ilave Giriş beklediğinde (insan işlevsel kullanıcıların iş uygulamaları yazılımlarına veri girişlerinde oluşabilir);
• bir işlevsel süreç, başlamasının ardından, işlevsel kullanıcıdan 'veriniz varsa şimdi iletin' mesajı ilettiğinde ve işlevsel kullanıcı mesajını ilettiğinde;

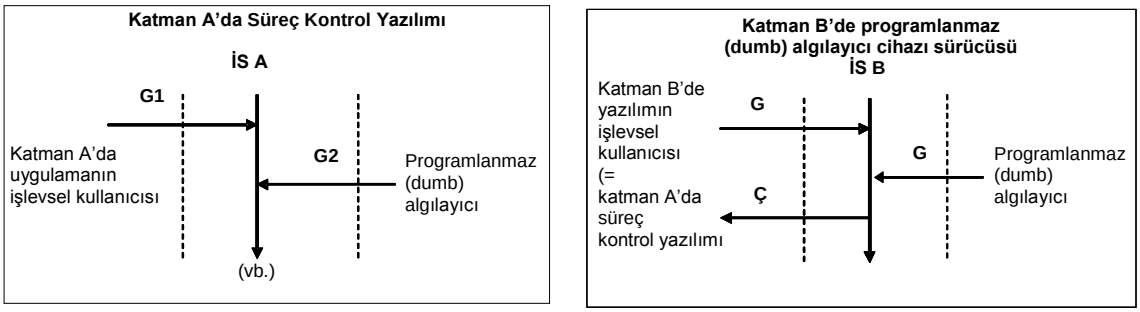
- bir işlevsel süreç, başlamasının ardından, işlevsel kullanıcının durumunu incelediğinde ve kendisine gereken veriyi aldığında.

  Son iki durumda (genellikle gerçek-zamanı 'tarama' (polling) yazılımlarında karşılaşılır) kural gereği gereken verinin alınmasında işlevsel süreçten bir Çıkış tanımlanmamalıdır. İşlevsel sürecin işlevsel kullanıcıya sadece bir ileti mesajı göndermesi gerekir ve bu ileti mesajının işlevselliği Girişin bir parçası olarak değerlendirilir. İşlevsel süreç nasıl bir veri beklediğini bilir. Bu durumda sadece bir Giriş gereklidir.

b) Bir işlevsel sürecin, bir işlevsel kullanıcının servisine ihtiyacı olduğu (örneğin veri edinme amacıyla) ve *işlevsel kullanıcının ne göndereceği hususunda bilgilendirilmesi gerektiği durumda* (genellikle işlevsel kullanıcının ölçülen yazılımın kapsamı dışında başka bir yazılım olduğu durumda), bir çift Giriş/Çıkış veri hareketi belirlenmelidir. Çıkış, özel veri isteğini; Giriş, geri döndürülen veriyi içerir.

ÖRNEK 1 (kural 'a)' bir ve ikinci maddeler için): Bir işlevsel süreç, bir insan işlevsel kullanıcısının veri girişi için, bir çevrim-içi iş uygulamasında olduğu gibi, biçimlendirilmiş ve bunun dışında olası ön değerler hariç 'boş' olan bir ekran sunduğu durumda, 'boş' ekranın sağlanması ayrı bir Çıkış olarak sayılmaz. Sadece doldurulan ekran bir veya birden çok Giriş olarak sayılır. (Ayrıca bakınız Kısım 4.1.10.)

ÖRNEK 2 (kural 'a)' üç ve dördüncü maddeler için): Bir gerçek-zamanlı süreç kontrol yazılım sisteminde bir işlevsel sürecin bir sıra özdeş programlanmaz (dumb) algılayıcıları taramak durumunda olduğunu varsayalım. İşlevsel süreç, verisini bir Giriş (tipi) aracılığıyla elde eder. (Birden çok olmalarına rağmen, algılayıcılar özdeş olduklarından tek bir Giriş (tipi) belirlenir ve sayılır.) Buna ek olarak, Girişin, Şekil 2.2.3.2'de gösterildiği üzere gerekli veriyi fiziksel olarak algılayıcıdan alan, pratikte yazılım mimarisinin daha alt katmalarındaki bir cihaz sürücü yazılımından geçirilmesi gerektiğini varsayalım. Süreç kontrol yazılım sisteminde ve programlanmaz algılayıcılardaki cihaz sürücü yazılımındaki işlevsel süreçler, aşağıda sunulan Şekil 4.1.9.1 (a) ve (b)'de gösterildiği gibi olur.



Şekil 4.1.9.1 (a) ve (b) Süreç kontrol uygulama katmanındaki yazılım 'A' tarafından programlanmaz cihaz sürücüsü katmanındaki yazılım 'B'ye bildirilen bir Giriş için Çözüm

Şekil 4.1.9.1 (a), süreç kontrol yazılımı işlevsel süreci 'İS A'nın, Giriş 'E1' ile, örneğin bir saat tıklaması tandan, tetiklendiğini göstermektedir. İşlevsel süreç daha sonra Giriş 'E2'yi, algılayıcı okumalarının çoklu oluşumlarını almak için programlanmaz algılayıcı sırasından elde eder. Programlanmaz algılayıcılar da süreç kontrol yazılımının işlevsel kullanıcılarıdırlar.

Şekil 4.1.9.1 (b), programlanmaz algılayıcıları süren yazılımın, 'İS B' işlevsel sürecinin tetikleyicisi olarak bir Giriş aldığını (muhtemel olarak pratikte işletim sistemi aracılığıyla) göstermektedir. Bu işlevsel süreç, kendi işlevsel kullanıcılarından (programlanmaz algılayıcı) algılayıcı verilerini elde etme amacıyla bir Giriş elde eder ve onu süreç kontrol yazılımına bir Çıkış olarak geri iletir. Daha sonra süreç kontrol yazılımının işlevsel süreci, kendi algılayıcı verisinin işletimi ile devam eder. Yine, her bir özdeş algılayıcıdan veri edinme döngüsünün birden çok olması, model açısından önemsizdir.

Süreç kontrol yazılımının Girişi ile cihaz sürücüsü yazılımının 'Giriş/Çıkış çifti' arasındaki belirgin uyumsuzluk, kural gereği bir Girişin herhangi bir 'okuma isteğini' kapsamasından kaynaklanmaktadır, ki; bu durumda işlevsel kullanıcının işlevsel süreçten gelen bir mesaja cevap verme yeteneği yoktur.

ÖRNEK 3 (kural 'b)' için): Bir işlevsel sürecin, 'akıllı' bir donanım cihazı veya herhangi bir eşdüzey yazılım gibi kendi işlevsel kullanıcılarından birine bir sorgu veya hesaplama için bazı parametreleri veya sıkıştırma için bazı verileri gönderdiğini varsayalım. İşlevsel kullanıcının yanıtı, kısım 4.1.8 Örnek 2'de açıklandığı gibi bir Giriş/Çıkış çifti aracılığıyla elde edilir.

4.1.10 Kontrol Komutu

'Kontrol komutu' sadece iş uygulamaları alanında geçerli olan ve işlevsel büyüklüğün ölçümünde göz ardı edilmesi gereken özel bir veri hareketi sınıfıdır. Tanımı şu şekilde olabilir:

TANIM – Kontrol Komutu
Bir kontrol komutu, bir işlevsel kullanıcının yazılım kullanımını kontrol etmesini sağlayan fakat bir ilgi nesnesi ile ilgili herhangi bir veri hareketi içermeyen bir komuttur. NOT: 'Kontrol komutu' terimi SADECE iş uygulamaları yazılımlarının ölçümü bağlamında kullanılmaktadır. Bu bağlamda, bir kontrol komutu bir veri hareketi değildir çünkü, bir ilgi nesnesi ile ilgili veri hareket ettirmez. Şu şekilde örnekler verilebilir: 'aşağı/yukarı sayfa' komutları, Tab veya Giriş tuşuna basılması, önceki bir eylemi onaylama amacıyla 'Tamam' tuşuna basılması, vb.

KURAL – İş uygulamaları alanında kontrol komutları
İş uygulamaları alanında 'kontrol komutları', bir ilgi objesi ile ilgili herhangi bir veri hareketi içermediğinden göz ardı edilmelidir.

ÖRNEKLER: İş uygulamaları alanında kontrol komutları işlevsel kullanıcının başlıkları veya hesaplanmış ara toplamların görünümünün kontrol edilmesine olanak veren işlevler, fiziksel ekranlar arasında ve bu ekranlarda aşağı-yukarı doğru gezinmesine sağlayan işlevler, bir hata mesajını veya her hangi bir veri girdisinin doğrulanmasını onaylamak amacıyla 'tamam' tuşuna basılması, vb. Dolayısıyla kontrol komutları ayrıca, işlevsel kullanıcının bir veya birden çok işlevsel süreç arasında gezinmesini sağlayan fakat kendileri bir işlevsel süreç başlatmayan menü komutlarını ve veri girişi için boş ekranların görüntülenmesini sağlayan komutları içerir.

DİKKAT: İş uygulamaları alanı dışında, 'kontrol komutu' kavramının özel bir anlamı yoktur ve bir ilgi objesi ile ilgili işlevsel kullanıcıdan gelen herhangi bir sinyal veya veri hareketi ölçülmelidir.

4.2 Ölçüm fonksiyonunun uygulanması

Bu basamak, her bir işlevsel süreç içerisinde tanımlanan her bir veri hareketine COSMIC ölçüm fonksiyonunun uygulanmasını içerir.

TANIM – COSMIC ölçüm fonksiyonu
COSMIC ölçüm fonksiyonu, COSMIC ölçüm standardına dayanarak fonksiyon değişkenlerine bir değer atayan matematiksel bir fonksiyondur. COSMIC ölçüm fonksiyonunun değişkeni veri hareketidir.

TANIM – COSMIC ölçüm standardı
COSMIC ölçüm standardı, 1 CİP (COSMIC İşlev Puan), bir veri hareketinin büyüklüğü olarak tanımlanmıştır.

Bu ölçüm fonksiyonuna göre; eklenmesi, değiştirilmesi veya silinmesi gerekli olan ve bölüm 4.1'e göre tanımlanmış veri hareketlerinin her bir örneği (Giriş, Çıkış, Okuma veya Yazma), 1 CİP sayısal büyüklüğünü alır.

4.3 Ölçüm sonuçlarının toplaması

Bu adım; tanımlanmış bütün veri hareketlerine uygulanarak bulunan ölçüm fonksiyonu sonuçlarının tek bir işlevsel büyüklük değerine toplanmasını içerir. Bu basamak aşağıdaki müteakip ilke ve kurallara göre tamamlanır.

4.3.1 Toplamanın genel kuralları

KURALLAR – Ölçüm sonuçlarının toplanması

a) Herhangi bir işlevsel süreç için, ayrı veri hareketlerinin işlevsel büyüklükleri aritmetik olarak eklenerek CİP birimi olarak toplanmalıdır.

Büyüklük (işlevsel süreç) = Σ büyüklük(Girişler$_i$) + Σ büyüklük(Çıkışlar$_i$) + Σ büyüklük(Okumalar$_i$)+Σ büyüklük(Yazmalar$_i$)

b) Herhangi bir işlevsel süreç için, İşlevsel Kullanıcı Gereksinimlerindeki değişimlerin işlevsel büyüklüğünü CİP birimi olarak vermek için işlevsel sürecin eklenen, değiştirilen veya silinen veri hareketleri aşağıdaki formüle göre toplanmalıdır.

Büyüklük (Değişim(işlevsel süreç $_i$)) = Σ büyüklük (eklenen veri hareketleri $_i$) + Σ size (değiştirilen veri hareketleri $_i$) + Σ size (silinen veri hareketleri $_i$)

İşlevsel büyüklüğün toplanması hakkında daha fazla bilgi için bakınız kısım 4.3.2. Değiştirilen yazılımın büyüklüğünün ölçümü için bakınız kısım 4.4.

c) Tanımlanmış bir kapsam içerisinde, bir yazılım parçasının büyüklüğünü bulmak için yazılım parçasının işlevsel süreç büyüklükleri, aşağıdaki e) ve f) kurallarına göre toplanmalıdır.

d) Tanımlanmış kapsam içerisinde, bir yazılım parçasındaki herhangi değişimin büyüklüğü, o parçaya ait bütün işlevsel süreçlerdeki değişimlerin aşağıdaki e) ve f) kurallarına göre toplaması ile elde edilir.

e) Katmanlar arasındaki yazılım parçasının büyüklüğü veya yazılım parçasındaki değişimin büyüklüğü, sadece İKG'leri aynı işlevsel süreç tanesellik (granularity) seviyesinde ölçüldü iseler toplanabilirler.

f) Herhangi bir katmandaki veya farklı katmanlardaki yazılım parçasının büyüklükleri ve/veya yazılım parçasındaki değişimlerin büyüklükleri sadece ölçümün amacı doğrultusunda toplanması anlamlı ise toplanabilir.

g) Bileşenler arası veri hareketlerinin toplam büyüklük ölçüsüne katkısı ortadan kaldırılmadıkça bir yazılım parçasının boyutu, onun parçalarını toplayarak elde edilmez (parçanın birleşenlerine nasıl ayrıştırıldığından bağımsız olarak).

h) COSMIC yönteminin yerel bir uzantısı oluşturulursa, (örneğin standart yöntemin içerisinde ele alınmamış bir büyüklük tarafını ölçmek amacıyla), yerel uzantı ile ölçülmüş olan büyülük 5.1'de tanımlandığı gibi ayrı bir şekilde rapor edilmeli ve standart yöntem ile elde edilmiş olan CİP birimindeki büyüklüğe eklenmemelidir (daha detaylı bilgi için bakınız kısım 4.5.)

Kural b) ve c) için ÖRNEK: Bir yazılım parçası için değişiklik istemi şu şekilde olabilir: '6 CİP büyüklüğünde yeni bir işlevsel süreç ekle, ve başka bir işlevsel süreçte bir veri hareketi ekle, ve üç başka veri hareketine geliştirmeler ekle ve iki veri hareketi sil.' Bu değişim isteminin toplam büyüklüğü şu şekildedir: 6+1+3+2= 12 CİP

Kural f) için ÖRNEK: Bir yazılım parçasının değişik kısımları farklı teknolojiler kullanılarak farklı proje takımları tarafından geliştirilmiş ise, bu parçaların büyüklüklerinin toplanması pratikte bir değer taşımayabilir.

Kural g) için ÖRNEK: Eğer bir yazılım parçası:

- ilk önce 'bir bütün olarak' ölçüldü ise, bir başka deyişle hepsi tek bir kapsam içerisinde;
- ikinci olarak parçaların büyüklükleri ayrı ayrı ölçüldü ise, bir başka deyişle her biri kendi kapsamı içerisinde,

bu durumda birleşenler arası veri hareketlerinin büyüklüğünden dolayı, ayrı ayrı ölçülmüş olan parçaların büyüklüklerini toplayarak bulduğumuz büyüklük (ikinci durum), bir bütün olarak ölçülmüş olan büyüklüğü (birinci durum) aşacaktır. Yazılım 'bir bütün olarak' ölçülmek istendiğinde bu birleşenler arası veri hareketleri görünmeyecektir ve bu nedenle büyüklüğü 'bir bütün' olarak elde etmek için birleşenler arası veri hareketleri göz ardı edilmelidir. 'İleri ve İlgili Başlıklar' dokümanındaki yalın yazılım mimarilerindeki değişken tanesellik seviyesi ölçümleri ile ilgili kısımdaki örnekleri de inceleyebilirsiniz.

Dikkat edilmelidir ki; belirlenmiş her bir katman *içerisinde*, toplama fonksiyonu tamamıyla ölçeklendirilebilir. Dolayısıyla, her bir ölçüm probleminin amacı ve kapsamı doğrultusunda ve yukarıdaki d), e) ve f) kuralları göz önünde bulundurularak, tekil işlevsel süreçler veya bir katman içerisinde yazılımın bütünü için bir alt toplam oluşturulabilir.

4.3.2 İşlevsel büyüklük toplamı hakkında daha fazla bilgi

İşlevsel büyüklüğün değişken olarak kullanıldığı bir modelde, örneğin işgücü tahmini için, ve ölçülecek yazılımın birden fazla katmanı veya eşdüzey birleşeni olduğu bir bağlamda, toplama genel olarak katman veya eşdüzey birleşen bazında gerçekleştirilecektir, çünkü bunlar genellikle aynı teknoloji kullanılarak geliştirilmemiştir.

ÖRNEK 1: Uygulama katmanı 3GL (3. kuşak programlama dili) ve mevcut kütüphaneler kümesi kullanılarak, sürücü katmanı ise makine dili kullanılarak geliştirilecek bir yazılım düşünün. Her bir katmanın oluşturulması için harcanacak olan birim işgücü büyük olasılıkla birbirinden farklı olacaktır ve sonuç olarak büyüklükleri baz alınarak her bir katmanın işgücü kestirimi ayrı ayrı hazırlanacaktır.

ÖRNEK 2: Eğer bir proje takımı birden fazla ana yazılım parçası geliştirmek durumunda ise ve takım, kendisinin toplam üretkenlik değeri ile ilgileniyorsa, her bir parçayı üretmek için gereken çalışma saatlerini toplayabilir. Aynı şekilde, bu büyüklükler yukarıda verilen kuralları karşılıyor ise proje takımı geliştirdiği ana yazılım parçalarının büyüklüklerini toplayabilir.

Aynı işlevsel süreç tanesellik seviyesinde ölçülmüş, standart katmanlı bir mimaride farklı katmanlardaki yazılım ana parçalarının büyüklüklerinin toplanabiliyor olmasının sebebi tutarlı şekilde tanımlanmış işlevsel kullanıcılarının var olmasıdır. Her bir katman, 'altındaki' katmanın bir işlevsel kullanıcısıdır ve bir katmandaki herhangi bir yazılım parçası aynı katmandaki başka bir eşdüzey yazılım parçasının işlevsel kullanıcısı olabilir. Böyle bir mimarinin gereksinimleri, farklı parçaların aralarında mesajlaşmaları gerektiğini ifade eder. Bu nedenle muhtelif yazılım parçalarının büyüklüklerinin toplanmasının mantıklı ve makul olması, yukarıda verilen d), e) ve f) kurallarına uygunluğuna bağlıdır. Ancak, tersine, yukarıda kural f)'de de belirtildiği üzere, objeler arası veri hareketleri ortadan kaldırılmadığı sürece ana yazılım parçasının büyüklüğü, o yazılım parçasının yeniden-kullanılabilir bileşenlerinin büyüklüklerinin toplanması ile elde *edilmeyebilir*.

Ölçüm sonuçlarının veri hareketlerinin tiplerine göre toplanması, her bir veri hareketi tipinin toplam büyüklüğe katkısının analizi açısından yararlı olabilir ve böylece ölçülen katmanın doğal işlevsel yapısının nitelenmesine yardımcı olabilir.

4.4 Yazılım değişikliklerinin büyüklüğünün ölçülmesi hakkında daha fazla bilgi

Mevcut bir yazılımdaki 'işlevsel değişiklik', COSMIC yöntemine göre 'yeni veri hareketlerinin eklenmesi, veya mevcut veri hareketlerinin değiştirilmesi veya silinmesinden oluşan herhangi bir birleşim' şeklinde değerlendirilir. Burada 'işlevsel değişiklik' olarak adlandırdığımız kavram için 'iyileştirme' ve 'bakım'[15] terimleri sıklıkla kullanılır.

[15] Normal ölçüm kuralları gereği, bir yazılımın bir hatayı düzeltmek amacıyla ve dolayısıyla kendi İKG'ne uygun şekle getirilmesi için değiştirilmesi gerekiyor ise, bu yazılım parçasının işlevsel büyüklüğü değişmez. Değişiklik bir yazılımın İKG'ndeki bir hatayı düzeltmek amacıyla yapılıyor ise, o yazılımın işlevsel büyüklüğü değişir.

Yazılımdaki bir değişiklik gereksinimi aşağıdaki herhangi bir sebepten kaynaklanıyor olabilir:

- yeni bir İKG (sadece mevcut işlevselliğe ilaveler), veya
- İKG'nde bir değişiklik (muhtemel olarak eklemeler, değişiklikler veya silmeler içeren), veya
- Bir hatayı düzeltmek amacıyla ihtiyaç duyulan 'bakım'.

Bu değişikliklerin herhangi birisinin ölçümü için geçerli kurallar aynıdır, ancak ölçümleyicinin performans ölçümleri veya kestirimleri yaparken çeşitli durumları ayırt etmesi önemlidir.

İşlevselliğini genişleterek ve/veya daraltarak veya bu şekilde olmaksınız bir yazılım parçasının yerine bütünüyle yenisi konulduğunda, örneğin yeniden geliştirme ile, bu değişimin işlevsel büyüklüğü, normal kurallara göre ölçülmüş yerine konulan yazılımın büyüklüğü kadardır. Bu durum, bu kısımda daha derin olarak ele alınmayacaktır. Ölçümleyici, performans ölçümleri veya kestirimleri yaparken tamamıyla baştan geliştirilen projeler ile mevcut bir yazılımın yeniden geliştirilmesi veya yerine yenisinin konulması projeleri arasındaki gereksinimlerin farkında olmalı ve ayrımı yapabilmelidir.

Bir uygulamanın kullanılmayan bölümü çoğunlukla kod parçası yerinde bırakılarak ve kullanılmayan işlevsellik ile ilişki kaldırılarak silinir ('koparılır' daha doğru bir tanımlama olacaktır). Kullanılmayan kısmın işlevsel büyüklüğü 100 CİP iken, kullanılmayan kısmın bağlantısı, örneğin 2 veri hareketinin değiştirilmesi ile kopartılabiliyor ise, işlevsel değişimin büyüklüğü 2 değil 100 veri hareketi olarak belirlenmelidir. Burada, geliştirilen kısmın büyüklüğü değil, gereksinimin büyüklüğü ölçülür[16].

İşlevsel değişimin büyüklüğü (burada tartışılan) ile yazılımın işlevsel büyüklüğündeki değişimin arasındaki farka dikkat ediniz. Genellikle, bunlar birbirinden farklıdır. Yazılımın işlevsel büyüklüğündeki değişim kısım 4.4.3'te anlatılacaktır.

4.4.1 İşlevselliği değiştirme

G, Ç, O veya Y tipinde herhangi veri hareketi iki çeşit işlevsellik içerir: tek bir veri grubu hareket ettirir ve ilgili bazı veri işlemeleri içerir (ikincisi için bakınız kısım 4.1.6). Dolayısıyla ölçüm amaçları doğrultusunda bir veri hareketinin değişmiş olması aşağıdaki durumlar oluşmuş ise düşünülebilir:

- Hareket ettirilen veri grubu ve/veya
- ilintili veri işlemesi,

herhangi bir şekilde değiştirilmiş ise.

Bir veri grubu aşağıdaki durumlarda değiştirilmiş olarak değerlendirilir:

- bu veri grubuna yeni öznitelikler eklenmiş ise ve/veya
- bu veri grubundan mevcut öznitelikler çıkarılmış ise ve/veya
- bir veya daha fazla öznitelik değiştirilmiş ise, örneğin anlamı veya biçimi (değeri değil).

Bir veri işlemenin işlevselliği herhangi bir şekilde, örneğin hesaplanması, özel biçimi, sunumu ve/veya veri doğrulaması değişikliğe uğramış ise, o veri işleme değiştirilmiş olarak kabul edilir. 'Sunum', yazı karakteri, arka plan rengi, alan uzunluğu, ondalık basamakların sayısı, vb. anlamlara gelebilir.

İlgi nesnelerine ilişkin veri hareket ettirmedikleri için kontrol komutları ve genel uygulama verileri, veri hareketi içermezler. Dolayısıyla, kontrol komutları ve genel uygulama verilerinde yapılan değişiklikler ölçülmemelidir. Örnek olarak; bütün ekranlar için ekran rengi değiştirildiğinde, bu değişiklik ölçülmemelidir. (Kontrol komutları ve genel uygulama verileriyle ilgili açıklamalar için bakınız kısım 4.1.10.)

---

[16] İşlevsellik koparma, 'gerçek' silmelerden oldukça farklı olduğundan, kestirimlerde, işlevsel değişikliğin bu kısmı için farklı üretkenlik değerlerinin kullanılması tavsiye edilir. Alternatif olarak, kestirimlerde, gereksinimin büyüklüğü yerine (verilen örnekte 100 CİP), uygulanan büyüklüğün (verilen örnekte 2 CİP) ölçülmesi tercih edilebilir. 'Proje büyüklüğü' 2 CİP olarak ölçüldüyse, bu açık şekilde belgelendirilmeli ve uygulamanın 100 CİP azaltılmasını isteyen İKG'nin ölçümünden ayrılmalıdır. '

KURALLAR - Bir veri hareketinin değiştirilmesi
a) Bir veri hareketine ilişkin veri işlemelerindeki bir değişiklikten ve/veya hareket ettirilen veri grubundaki öznitelik sayısındaki veya tipindeki değişiklikten dolayı bir veri hareketi değiştirilmek durumunda ise; tek bir veri hareketinde yapılan değişikliklerin asıl sayısından bağımsız olarak değişiklik 1 CİP olarak ölçülmelidir. b) Bir veri grubunun değiştirilmesi gerekiyor ise, işlevselliği bu değişiklikten etkilenmemiş olan ve değiştirilmiş veri grubunu hareket ettiren veri grubunu taşıyan veri hareketleri, değiştirilmiş veri hareketleri olarak tanımlanmamalıdır. NOT 1: Özniteliğin değerindeki bir değişiklik, örneğin değerleri kodlama sistemi olan bir özniteliğin tekil kod değerinde yapılan bir değişiklik, özniteliğin tipinde yapılmış bir değişiklik değildir. NOT 2: Bir İşlevsel kullanıcıya göre bir ilgi nesnesi ile ilişkili olmayan ve giriş veya çıkış ekranlarında beliren herhangi bir verideki değişiklik, değişmiş bir CİP olarak değerlendirilmemelidir (bu gibi veriler için bakınız kısım 3.3.4).

Kural a) ve b) için ÖRNEK: Bir veri grubunun (D_1) veri özniteliklerini ekleme veya değiştirme yaparak o veri grubunu D_2 haline getirecek bir gereksinim olduğunu varsayalım. Bu değişikliğin yapılacağı işlevsel süreç 'A'da, değişiklikten etkilenen tüm veri hareketleri belirlenmeli ve 'değiştirilmiş' olarak sayılmalıdır. Buna göre, Kural a) gereği, eğer değişmiş veri grubu D_2 kalıcı hale getirilmiş ve/veya işlevsel süreç A'nın çıktısı ise, sırasıyla bir Yazma ve/veya bir Çıkış veri hareketi 'değiştirilmiş' olarak belirlenmelidir. Ancak, aynı veri grubu D_2'yi Okuyan veya Giriş yapan diğer işlevsel süreçler de olabilir; fakat bu işlevsel süreçler değiştirilmiş veya eklenmiş veri özniteliklerini kullanmadıkları için onların işlevselliği bu değişiklikten etkilenmez. Bu işlevsel süreçler, hareket ettirilen veri gurubunu D_1 olarak işlemeye devam ederler. Buna güre Kural b) gereği, işlevsel süreç 'A'nın veri hareketlerinde yapılan değişiklikten(lerden) etkilenmeyen diğer işlevsel süreçlerdeki veri hareketleri, değiştirilmiş olarak *belirlenmemeli* ve *sayılmalıdır*.

4.4.2 İşlevsel olarak değiştirilen yazılımın büyüklüğü

Bir yazılım parçasını işlevsel olarak değiştirdikten sonra, yeni toplam işlevsel büyüklük için, orijinal büyüklüğüne, eklenen tüm veri hareketlerinin işlevsel büyüklüğü eklenmeli, kaldırılan tüm veri hareketlerinin işlevsel büyüklüğü çıkarılmalıdır. Değiştirilmiş veri hareketlerinin yazılım parçasının büyüklüğüne etkisi yoktur, çünkü bu veri hareketleri değişiklikten önce de ve sonra da varlıklarını sürdürürler.

4.5 COSMIC Ölçme Yönteminin Genişletilmesi

4.5.1 Giriş

İşlevsel büyüklüğü ölçme yöntemi COSMIC, yazılım büyüklüğünün tüm yönlerini ölçmeyi öngörmemektedir. Mevcut durumda COSMIC ölçme yöntemi, karmaşık matematiksel algoritmaları ya da karmaşık kuralları içeren uzman sistemler gibi belirli tipteki işlevsel kullanıcı gereksinimleri çeşitleri için standart bir hesaplama yolu sağlamak üzere tasarlanmamıştır. Bunun yanında, her veri hareketi için veri özniteliği sayısının işlevsel büyüklüğe etkisi bu yöntem tarafından dikkate alınmaz. Veri işlenmesi alt sürecinin büyüklüğünün etkisi, Bölüm 2.1, Yöntemin Uygulanabilirliği kısmında anlatıldığı şekilde, belirli yazılım alanları için geçerli olan varsayımları sadeleştirmek yoluyla hesaba katılmaktadır.

"Karmaşıklık" gibi diğer parametrelerin işlevsel büyüklüğe katkısı söz konusu olabilir. Bu hususta yapıcı tartışmalar öncelikle diğer parametrelerin, halen tam olarak tanımlanamamış yazılım büyüklüğü kavramı bağlamında, genel olarak kabul görmüş tanımları olmasını gerektirmektedir. Bu tanımlar, şu an için halen araştırma ve tartışma konusudur.

Buna rağmen, COSMIC büyüklük ölçüsü yöntemlerin belirlenen amaçları ve uygulama alanları için iyi bir yaklaşım olarak düşünülmektedir. Organizasyonların lokal ortamları içinde, COSMIC ölçme yöntemi kullanılarak organizasyona özgü anlamlı bir lokal standart tanımlanmak istenebilir. Bu

nedenle, COSMIC Ölçme Yöntemi lokal ortamlar için genişletilmeye uygundur. Bu lokal uzantılar kullanıldığında, ölçme sonuçları Bölüm 5.1.'de sunulan özel kurallara göre raporlanmalıdır.

Bundan sonraki bölümler yöntemin lokal standartlar ile nasıl genişletilebileceğini göstermektedir.

4.5.2 Karmaşık algoritmalar için lokal uzantılar .

Karmaşık algoritmalar için hesaplama gerekliyse, bu istisna işlevsellik için bir lokal standart geliştirilebilir. Anormal karmaşık veri işleme alt süreci içeren herhangi bir işlevsel süreçte, ölçüm yapan kişi kendi lokal işlev puanı birimini atamakta serbesttir.

ÖRNEK: Bir lokal uzantıya örnek şu şekilde olabilir: "Bizim organizasyonumuzda, matematiksel algoritmalar için atanan bir lokal işlev puanı … (anlamlı ve anlaşılabilir örnekler listesi olabilir) karşılık gelir….vs…

İki lokal işlev puanı ….. (diğer örnekler listesi) karşılık gelir…vs"

4.5.3 Ölçmenin alt birimleri için lokal uzantılar

Veri hareketlerinin ölçümünde daha çok kesinlik gerektiğinde, ölçümün alt birimleri tanımlanabilir. Örneğin, metre 100 santimetreye ya da 1000 milimetreye bölünebilir. Benzeşimle, bir veri özniteliğinin hareketi, ölçümün alt birimleri olarak kullanılabilir. COSMIC ile örnek yazılımlarda yapılan alan testleri ölçümleri, her veri hareketi için ortalama veri özniteliği sayısının dört veri hareketi tipi için çok fazla değişmediğini göstermiştir. Bu nedenle ve ölçüm sonuçlarının kolaylığı için, COSMIC ölçüm birimi, 1 CİP (COSMIC İşlev Puan), bir veri hareketi için sabitlenmiştir. Buna rağmen, ortalama veri özniteliği sayısının her veri hareketi tipi için keskin farklılık göstereceği, yazılımın farklı iki parçasının CİP cinsiden ölçülen büyüklüğü karşılaştırıldığında açıkça uyarı gereklidir.

COSMIC yöntemini iyileştirmek isteyen biri, ölçümün alt birimlerini tanımlayarak bunu yapabilir fakat sonuç büyüklük ölçümü standart COSMIC İşlevsel Puanı ile ifade edilmemelidir.

5

ÖLÇMENİN RAPORLANMASI

Genel Yazılım Modeli satırların işlevsel süreçleri (katmanlar şeklinde gruplanabilirler), sütunların veri gruplarını ve hücrelerin belirlenen alt-süreçleri (Giriş, Çıkış, Okuma ve Yazma) ifade ettiği bir matris şeklinde gösterilebilir. Genel Yazılım Modeli'nin bu şekildeki gösterimi Ek A'da verilmektedir.

COSMIC ölçümü sonuçları raporlanacağı ve arşivleneceği zamana aşağıdaki geleneklere uyulmalıdır.

5.1 Etiketleme

Bir COSMIC büyüklük ölçümü raporlanacağı zaman ISO/IEC 14143-1: 2007 standardına uygun olarak aşağıdaki geleneğe göre etiketlenir.

KURAL – COSMIC ölçme etiketleme
Bir COSMIC ölçme sonucu "***x*** CİP (v.***y***) " olarak ifade edilir. Burada, • "**x**" işlevsel büyüklüğün sayısal değerini ifade eder, • "v.**y**" sayısal işlevsel büyüklük değeri "x"'i elde etmede kullanılan standart COSMIC yöntemi sürümünü ifade eder. NOT: Eğer ölçme için yerel yaklaştırma yöntemi kullanılmış, ancak ölçme bir standart COSMIC sürümü kurallarına göre yapılmışsa, yukarıdaki etiketlendirme geleneği kullanılır, ancak yaklaştırma yöntemi bir yere not edilir – bölüm 5.2'ye bakınız.

ÖRNEK: Bu kılavuzdaki kuralları uygulayarak elde edilmiş bir ölçüm sonucu 'X CİP (v.3.0.1)' olarak gösterilir.

Eğer yukarıdaki bölüm 4.5'de tanımlandığı gibi yerel genişletilmeler kullanılmışsa, ölçme sonucu aşağıdaki gibi raporlanır.

KURAL – COSMIC yerel genişletmeler etiketlendirmesi.
Yerel genişletmeler kullanılarak elde edilen bir COSMIC ölçme sonucu şu şekilde ifade edilir: " ***x*** CİP (v. ***y***) + z Yerel İP", burada: • "**x**" standart COSMIC yöntemi sürüm v.y'ye göre yapılan her bir ölçme sonucunun toplanması ile elde edilen sayısal değeri ifade eder. • "v.**y**" sayısal işlevsel büyüklük değeri "x"'i elde etmede kullanılan standart COSMIC yöntemi sürümünü ifade eder. • "z" standart COSMIC yönteminin yerel genişletmelerine göre yapılan her bir ölçme sonucunun toplanması ile elde edilen sayısal değeri ifade eder.

5.2 COSMIC ölçme sonuçlarının arşivlenmesi.

COSMIC ölçme sonuçları arşivlenirken, sonucun her zaman anlaşılabilir olmasını sağlamak amacı ile aşağıdaki bilgi de tutulmalıdır.

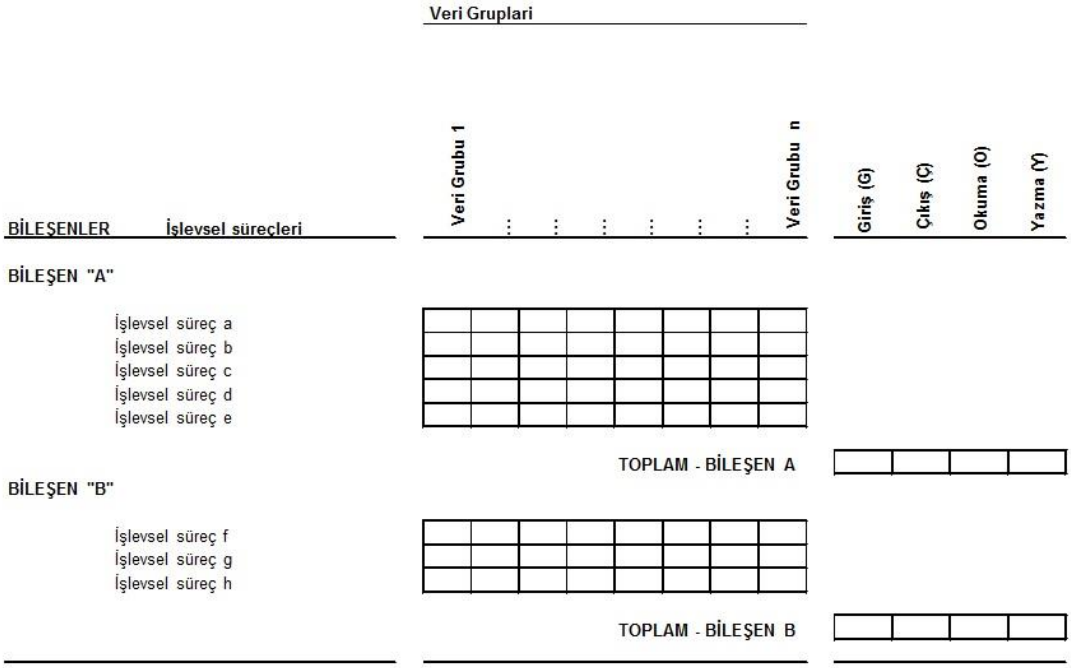
KURAL– COSMIC ölçme raporlama

5.1'deki gibi kaydedilmiş gerçek ölçümlere ek olarak, her ölçümle ilgili aşağıdaki öznitelikler de kayıt edilmelidir.

a) Ölçülen yazılım bileşeninin tanımlanması (adı, sürüm No veya konfigürasyon No).
b) Ölçme için kullanılan İKG'nin belirlenmesi için kullanılan bilgi kaynaklar.
c) Yazılımın alanı.
d) Ölçmenin amacını içeren ifade.
e) Ölçüm kapsamının tanımı ve varsa diğer ilişkili ölçümlerin toplam kapsamıyla olan ilişkisi. (Bölüm 2.2'de verilen genel kapsam kategorilerini kullanınız)
f) Yazılımın İşlevsel Kullanıcıları.
g) İKG'nin tanesellik seviyesi ve yazılımın ayrıştırma seviyesi.
h) Proje yaşam döngüsü içinde ölçümün gerçekleştirildiği noktanın tanımı. (Ölçümün tamamlanmamış İKG'ler üzerinden gerçekleştirilen bir kestirim mi olduğu yoksa teslim edilen işlevsellik üzerinden gerçekleştirilen bir ölçüm mü olduğu belirtilmelidir.)
i) Ölçümün hedeflenen veya varsayılan hata payı.
j) Standart COSMIC yöntemi kullanılmışsa veya yönteme herhangi bir yereli yaklaşım getirilmişse ve/veya yönteme yerel genişletmeler uygulanmışsa bunların belirtilmesi. (Bölüm 4.5'e bakınız). Bölüm 5.1 ve 5.2'de verilen etiketleme standartlarını kullanınız.
k) Ölçümün geliştirilen işlevsellik mi teslim edilen işlevsellik için mi olduğunun belirtilmesi. ("Geliştirilen işlevsellik" yeni geliştirilen yazlımın büyüklüğünü ifade eder; "teslim edilen işlevsellik" geliştirilen işlevselliği içermekle beraber, yeni geliştirme dışındaki yollarla elde edilen işlevselliği de kapsar. (örn: her türlü tekrar kullanım, var olan işlevselliği ekleyip çıkarmak için kullanılan parametreler vb.)
l) Ölçümün yeni oluşturulan işlevsellik için mi yoksa bir bakım/değişiklik etkinliği sonucunda mı gerçekleştirildiğinin belirtilmesi. (Eklenen, değiştirilen ve çıkartılan işlevselliğin toplamı. Bkz. 4.4)
m) Varsa, ölçümün yapıldığı katmanlara ait mimarinin tanımı.
n) Varsa, toplam büyüklüğü oluşturmak için büyüklükleri toplanan ana bileşenlerin sayısı.
o) Toplam ölçüm kapsamı içindeki her kapsam için, Ek-A'da belirtildiği gibi bir ölçüm matrisi.
p) Ölçümü yapan kişinin adı ve sahip olduğu COSMIC sertifikasyon dereceleri.

EK A – COSMIC BÜYÜKLÜK ÖLÇÜMÜNÜ BELGELEME

Aşağıdaki yapı, Genel Yazılım Modeline eşlenmiş bütün bir kapsamdan tespit edilen her bir bileşene ait sonuçları tutmak için kaynak olarak kullanılabilir. Bütün ölçüm kapsamı içerisindeki her bir kapsamın kendi matrisi vardır.



Şekil A – Genel Yazılım Modeli matrisi

EŞLEME AŞAMASI

- Tespit edilen her bir veri grubu bir sütun olarak kaydedilir.
- Tespit edilen her bir işlevsel süreç, tespit edilmiş olan bileşen ile gruplanmış olarak spesifik bir satıra kaydedilir.

ÖLÇME AŞAMASI

- Tespit edilen her bir işlevsel süreçte bulunan veri hareketleri ilgili hücrelere aşağıdaki konvansiyona uygun şekilde not edilir: Giriş için “G”, Çıkış için “Ç”, Okuma için “O” ve Yazma için “Y”.
- Daha sonra, tespit edilen her bir işlevsel süreç için veri tipi bazında toplamlar hesaplanır ve matrisinin en alt satırındaki ilgili sütuna kaydedilir.
- Daha sonra da ölçüm özeti, her bir bileşenin “TOPLAM” satırındaki kutu içine alınmış hücreye yazılabilir.

EK B – COSMIC İLKELERİ ÖZETİ

Aşağıdaki tablo, birebir karşılaştırma COSMIC kılavuzunda bulunan bütün ilkeleri sıralar.

NO	İLKE TANIMI
İ-01	**COSMIC Yazılım Bağlam Modeli** a) Yazılım donanım tarafından sınırlandırılmıştır. b) Yazılım genel olarak **katmanlardan** oluşmaktadır. c) Bir katman, bir ya da daha fazla birbirinden farklı **eş (peer)** yazılım parçası içerebilir ve bu yazılım parçaları da ayrı eş bileşenlerden oluşabilir. d) Ölçülecek yazılım parçası, ölçüm **kapsamı** tarafında belirlenmeli ve kapsam tek bir katman ile sınırlı olmalıdır. e) Ölçülecek yazılımın kapsamı ölçümün **amacına** bağlı olarak belirlenmelidir. f) Yazılımdan veri alan ya da yazılıma veri gönderen **işlevsel kullanıcılar**, yazılımın işlevsel kullanıcı gereksinimlerinden yola çıkarak belirlenmelidir. g) Yazılım, işlevsel kullanıcıları ile **sınırı** geçen **veri hareketleri** ve sınır içerisinde yer alan kalıcı bellek alanına veri gönderen ya da kalıcı bellek alanından veri alan yazılım parçası aracılığıyla etkileşimde bulunur. h) Yazılımın İKG'leri farklı **detay seviyelerinde** verilmiş olabilir. i) Ölçümün yapılacağı tanesellik seviyesi normal olarak işlevsel süreçler seviyesinde olmalıdır. j) Eğer işlevsel süreçler bazında ölçüm yapmak mümkün değilse, yazılımın İKG'leri kestirim yaklaşımı ile ölçülmeli ve işlevsel süreçler tanesellik seviyesinde ölçeklenmelidir[4].
İ-02	**COSMIC Genel Yazılım Modeli** a) Yazılım, işlevsel kullanıcılarından **girdi** verisini alır ve işlevsel kullanıcıları için **çıktı** verisini üretir. b) Ölçülecek yazılımın işlevsel kullanıcı gereksinimleri birbirinden ayrı ve eşsiz işlevsel süreçlerle eşleştirilebilir. c) Her işlevsel süreç **alt-süreçlerden** oluşmaktadır. d) Her bir alt-süreç ya bir **veri hareketi** ya da bir **veri işlemedir**. e) Her işlevsel süreç bir işlevsel kullanıcının belirlediği bir olay sonrasında işlevsel kullanıcı tarafından bir **Girdi** veri hareketi ile tetiklenir. f) Bir veri hareketi tek bir **veri grubunu** hareket ettirir. g) Bir veri grubu tek bir ilgi nesnesini tarif eden eşsiz **veri öznitelik** setinden oluşur. h) Dört tip veri hareketi vardır. **Girdi**, bir veri grubunu işlevsel kullanıcıdan yazılıma doğru hareket ettirir. **Çıkış**, bir veri grubunu yazılımdan işlevsel kullanıcıya doğru hareket ettirir. **Yazma**, bir veri grubunu yazılımdan kalıcı belleğe doğru hareket ettirir. **Okuma**, bir veri grubunu kalıcı bellekten yazılıma doğru hareket ettirir. i) Bir işlevsel süreç en az bir Girdi veri hareketi veri a bir Yazma ya da bir Çıkış veri hareketi içermelidir. Bu nedenle bir işlevsel süreç en az iki veri hareketinden oluşur.

NO	İLKE TANIMI
	j) Ölçüm sürecinin bir varsayımı olarak, veri işleme alt-süreçleri ayrı olarak sayılmaz; veri işleme alt-süreçlerinin işlevselliklerinin ilişkili oldukları veri hareketi tarafından hesaba katıldığı varsayılmaktadır.
İ-03	**COSMIC Ölçüm İlkesi** Bir yazılım parçasının işlevsel büyüklüğü, içerdiği veri hareketi sayısıyla doğru orantılıdır.
İ-04	**Katman** a) Bir katmandaki yazılım diğeriyle kendi işlevsel süreçleri aracılığı ile veri alış verişinde bulunur. b) Katmanlar arasındaki hiyerarşi ilişki, bir katmanın hiyerarşide altında olan katmanların işlevsel hizmetlerini kullanacağı şekildedir. k) Bu şekilde kullanım ilişkilerinin olduğu durumlarda, kullanan katmanı üst, kullanılan katmanı alt olarak tanımlıyoruz. Üst katmandaki yazılım, düzgün çalışabilmesi için alt katmanlardaki yazılıma hizmetlerine güvenir; alttakiler aynı şekilde düzgün çalışabilmek için kendi altındaki katmanlara güvenir ve hiyerarşide aşağı indikçe bu şekilde devam eder. Bu durumun tersine, bir alt katman bağımlı olduğu alt katmanlarla beraber hiyerarşideki üst katmanlardaki yazılım hizmetlerine ihtiyaç duymadan çalışabilir. c) Bir katmandaki yazılım, yazılımın alt katmalarda sağlanan servislerin tümünü kullanmak zorunda değildir. d) İki farklı katmadaki yazılım arasında alınıp verilen veri, her katmanın kendi İKG'ine göre farklı şekilde tanımlanıp yorumlanabilir. Yani, iki yazılım parçası, alınıp verilen verinin farklı veri öznitelikleri ve veri alt gruplarını tanıyabilir. Yine de, alıcı katmanın verici katmandan gelen veriyi yorumlayabilmesi için ortak olarak tanınan, alıcı yazılımın ihtiyaçlarına göre belirlenen, bir ya da daha fazla veri özniteliği ya da alt grubu olmalıdır.
İ-05	**Eş Bileşen** a) Bir yazılım parçasının bir katmanında bulunan eş bileşenler kümesi arasında, katmanlarda olduğu gibi bir hiyerarşik ilişki yoktur. Bir yazılım parçasının aynı katmandaki tüm eş bileşenlerinin İKG, katman hiyerarşisinde aynı seviyede bulunur. b) Yazılım parçasının başarıyla çalışması için tüm eş bileşenleri karşılıklı olarak beraber çalışmalıdır. c) Herhangi bir katmanda, ortak taşınan ya da paylaşılan verisi olan tüm eş bileşen İKG, bu veriyi aynı şekilde tanımlamalı; ortak taşınan yada paylaşılan verinin aynı özniteliklerini ve veri gruplamalarını tanımalıdır.
İ-06	**COSMIC Genel Yazılım Modelinin Uygulanması** COSMIC Genel Yazılım Modeli ayrı ölçme çerçevesi tanımlanan her bir ayrı yazılımın işlevsel kullanıcı gereksinimlerine uygulanacaktır. 'COSMIC Genel Yazılım Modelinin Uygulanması' demek İKG'nde belirlenen ve her bir kullanıcı (tipleri) tarafından fark edilen tetikleyici olaylar setinin belirlenmesi, ve daha sonra bu olaylara karşılık vermek üzere sağlanması gereken karşılık gelen işlevsel süreçlerin, ilgilenilen nesnelerin, veri gruplarının ve veri hareketlerinin belirlenmesidir.

NO	İLKE TANIMI
İ-07	**Veri Grubu** a) Her bir belirlenen veri grubu kendisinin tek olan veri öznitelikleri topluluğu yoluyla tek ve ayrılabilir olacaktır. b) Her bir veri grubu yazılımın İşlevsel Kullanıcı Gereksinimlerindeki herhangi bir ilgi nesnesi ile direk olarak ilişkili olacaktır. c) Bir veri grubu yazılımı destekleyen bir bilgisayar sisteminde gerçekleştirilecektir.
İ-08	**Giriş (G)** a) Bir Giriş tek bir ilgi nesnesini tanımlayan bir veri hareketini işlevsel kullanıcıdan uygulama sınırından kendisinin de bir parçası olduğu işlevsel sürece doğru hareket ettirmelidir. Bir işlevsel sürecin girdisi birden çok veri grubu içeriyor ise, girdideki her bir benzersiz veri grubu için bir Giriş tanımlanır (ayrıca bakınız kısım 4.1.7 - 'Veri Hareketi Benzersizliği'). b) Bir Giriş uygulama sınırından bir veriyi çıkaramaz, veriyi okuyamaz veya yazamaz. c) Bir işlevsel sürecin işlevsel kullanıcıdan veri alması gerektiği, fakat kullanıcıya hangi verinin gönderileceğinin söylenmesinin gerekli olmadığı veya işlevsel kullanıcının gelen herhangi bir mesaja tepki veremeyeceği durumda, verinin alınması için işlevsel süreçte bir Giriş belirlenir. Bu durumlarda, işlevsel süreçten işlevsel kullanıcıya verinin elde edilmesi için gönderilen herhangi bir mesaj bir Çıkış olarak sayılmamalıdır. Ancak, işlevsel sürecin işlevsel kullanıcıdan veri alması gerektiği ve işlevsel sürecin talebi karşılamak amacıyla işlevsel kullanıcıya veri temin etmesi gerektiği durumda, talep için bir Çıkış ve talep edilen verinin karşılığı olarak bir Giriş belirlenir (daha fazla bilgi için bakınız kısım 4.1.9).
İ-09	**Çıkış (Ç)** a) Bir Çıkış, tek bir ilgi nesnesini tanımlayan bir veri hareketini kendisinin de bir parçası olduğu işlevsel süreçten uygulama sınırından işlevsel kullanıcıya doğru hareket ettirir. Eğer bir işlevsel süreçten çıktılar birden fazla veri grubundan oluşuyorsa, çıktıdaki benzersiz her bir veri grubu için bir Çıkış belirlenir (ayrıca bakınız kısım 4.1.7 - 'Veri Hareketi Benzersizliği'). b) Bir Çıkış, uygulama sınırından bir veriyi geçiremez, bir veriyi okuyamaz veya yazamaz.
İ-10	**Okuma (O)** a) Bir Okuma, tek bir ilgi nesnesini anlatan tek bir veri grubunu kalıcı bellekten, Okumanın bir parçasını oluşturduğu işlevsel sürece hareket ettirmelidir. Bir işlevsel süreç kalıcı bellekten birden çok veri grubu getirmek durumunda ise, getirilen her bir benzersiz veri grubu için bir Okuma tanımlanır (ayrıca bakınız kısım 4.1.7 - 'Veri Hareketi Benzersizliği'). b) Bir Okuma, sınır dışına veri çıkarmamalı, sınır dışından veri almamalı veya veri yazmamalıdır. c) Bir işlevsel süreç sırasında, işlevsel sürecin dahilinde gerçekleşen ve sadece programcı tarafından değiştirilebilir sabit veya değişkenlerin hareket ettirilmesi veya işletilmesi; veya İKG'nden ziyade sadece uygulamadan kaynaklanan, ara sonuçların veya işlevsel sürecin depoladığı verilerin hesaplanması, Okuma veri hareketi olarak değerlendirilmemelidir.

NO	İLKE TANIMI
	d) Bir Okuma veri hareketi her zaman 'Okuma isteği' işlevini içerir (bu yüzden, 'okuma isteği' işlevi hiçbir zaman ayrı bir veri hareketi olarak sayılmamalıdır). Ayrıca bakınız kısım 4.1.9.
İ-11	**Yazma (Y)** a) Bir Yazma, tek bir ilgi nesnesini anlatan tek bir veri grubunu, Yazmanın bir parçasını oluşturduğu işlevsel süreçten, kalıcı belleğe hareket ettirmelidir. Bir işlevsel süreç, kalıcı belleğe birden çok veri grubu hareket ettirmek durumunda ise, kalıcı belleğe hareket ettirilen her bir benzersiz veri grubu için bir Yazma tanımlanır (ayrıca bakınız kısım 4.1.7 - 'Veri Hareketi Benzersizliği'). b) Bir Yazma, sınır dışından veri almamalı, sınır dışına veri çıkarmamalı veya veri okumamalıdır. c) Kalıcı bellekten bir veri grubunun silinmesine ilişkin bir gereksinim, tek bir Yazma veri hareketi olarak ölçülmelidir. d) Bir işlevsel süreç sırasında, işlevsel süreç tamamlandığında kalıcı olmayan verinin hareket ettirilmesi veya işletilmesi; işlevsel sürecin dahilinde olan değişkenlerin güncellenmesi veya hesaplamalarda ara sonuçların üretilmesi Yazma veri hareketi olarak değerlendirilmemelidir.
İ-12	**Veri Hareketleri ile ilişkili veri işleme** a) Bir işlevsel süreçteki tüm veri işlemeleri dört veri hareketi tipi (G,Ç,O,Y) ile ilişkilendirilmelidir. Kural gereği, bir işlevsel sürecin veri hareketlerinin, işlevsel sürecin veri işlemelerini de temsil ettiği varsayılır.

EK C – COSMIC ÖLÇÜM KURALLARININ ÖZETİ

Aşağıdaki tablo, birebir karşılaştırma COSMIC kılavuzunda bulunan bütün kuralları sıralar.

NO	KURAL TANIMI
K-01	**Kapsam** a) Bir İşlevsel Büyüklük Ölçümünün kapsamı, ölçüm amacından türetilmelidir. b) Bir ölçümün kapsamı ölçülecek yazılımın birden fazla katmanını içermemelidir.
K-02	**Katman** a) Yazılım burada bahsedildiği tipte bir katmanlar mimarisi kurgusu üzerinden tanımlanmışsa, bu mimari, ölçüm amacıyla katmanların belirlenmesi için kullanılmalıdır. b) BYS ya da iş yazılımları alanında, tepe katman, başka bir katmanın altı olmayan, normalde uygulama katmanı olarak adlandırılır. Bu katmandaki yazılım (Uygulama) düzgün çalışabilmek için diğer tüm katmanlardaki yazılım servislerine bağlıdır. Gerçek zamanlı yazlım alanında, tepe seviye, süreç kontrol yazılım sistemi, uçuş kontrol sistemi yazılımı örneklerindeki gibi, genelde sistem olarak adlandırılır. c) Emin olunamadığında, bağımlılık ve bağıntılılık kavramlarını etkileşen katmanları ayrıştırmak için kullanın. (ek-D ye bakınız) d) Mimari tasarım ve yapı göz önüne alınmadan evrilmiş bir yazılımın COSMIC modeline göre katmanlara ayrılabileceğini kabul etmeyin.
K-03	**İşlevsel Kullanıcılar** a) Ölçülecek bir yazılım parçasının işlevsel kullanıcıları, ölçüm amacından türetilmelidir. b) Bir yazılımın ölçüm amacı yazılım parçasının geliştirilmesi ya da değiştirilmesi için gerekli olan işgücü ile ilgili olduğunda, işlevsel kullanıcılar, yeni ya da değiştirilmiş işlevsellik sağlananlardır.
K-04	Sınır a) Ölçülen yazılım ile etkileşen işlevsel kullanıcıları belirle. Sınır işlevsel kullanıcılar ile yazılım arasındadır. b) Tanım gereği, bir katmandaki yazılım ölçülecek diğer katmandaki yazılımın işlevsel kullanıcısı olmak üzere belirlenmiş her bir katman çifti arasında bir sınır vardır. Benzeri şekilde, aynı katmandaki herhangi iki eş bileşen arasında bir sınır vardır; bu durumda her bir bileşen eşinin işlevsel kullanıcısı olabilir.
K-05	**İşlevsel süreç tanesellik seviyesi** a) İşlevsel büyüklük ölçümü işlevsel süreç tanesellik seviyesinde yapılmadır. b) Henüz, tüm işlevsel süreçlerinin tanımlanmadığı, veri hareketlerinin detaylandırılmadığı bir İKG'nin işlevsel büyüklük ölçümü gerektiğinde, ölçümler tanımlanan işlevsellik üzerinden yapılıp daha sonra işlevsel süreç tanesellik seviyesine ölçeklendirilmelidir. (İşlevsel büyüklüğü İKG'ni oluşturulma sürecinde erkenden kestirebilmek gibi yaklaşık büyüklük ölçme metotlarına "İleri ve İlgili Başlıklar" dokümanından bakınız.)
K-06	**İşlevsel Süreç** a) Bir işlevsel süreç anlaşılan kapsamdaki en az bir belirlenebilir İşlevsel

NO	KURAL TANIMI
	Kullanıcı Gereksiniminden türetilecektir. b) Bir işlevsel süreç (-tipi) belirlenebilir bir tetikleyici olay (-tipi) olunca gerçekleştirilecektir. c) Belli bir olay (-tipi) paralel olarak çalışan bir veya birden fazla işlevsel süreç (-tipi) tetikleyebilir. Belli bir işlevsel süreç (-tipi) bir veya birden fazla olay (-tipi) tarafından tetiklenebilir. d) Bir işlevsel süreç en azından iki veri hareketi içerecektir, bir Giriş ve bir Çıkış veya Yazdırma. e) Bir işlevsel süreç sadece ve sadece bir yazılım katmanına ait olacaktır. f) Gerçek-zamanlı yazılım bağlamında, bir işlevsel süreç kendi tarafından oluşturulan bir bekleme durumuna girdiğinde (yani işlevsel süreç tetikleyici olaya karşı yapması gereken her şeyi yapıp bir sonraki tetikleyici Girişi alana kadar beklemeye başladığında) sonlanmış kabul edilecektir. g) Ait olduğu İKG bir işlevsel sürecin maksimum sayıdaki girdi veri özniteliklerinden oluşan değişik alt-kümeleriyle oluşmasına izin verse bile ve bu değişimler ve/veya değişen girdi değerleri işlevsel süreçte değişik işlemsel patikalara yol açsa bile bir işlevsel süreç (-tipi) tanımlanacaktır. h) Aşağıdaki durumlarda ayrı olay (-tipleri) ve bunun sonucunda da ayrı işlevsel süreç (-tipleri) ayırt edilmelidir: i) Kararlar zaman içinde salıverilen ayrı olaylara yol açtığında, (örneğin, emir verisini bugün girmek ve daha sonra emrin kabulünü onaylamak ayrı bir karar gerektirdiğinden iki ayrı işlevsel sürecin belirtilmesi olarak değerlendirilmelidir), j) Aktiviteler için sorumluluklar ayrıldığında (örneğin, bir personel sisteminde temel kişisel verilerin idamesinin sorumluluğunun bordro verilerinin idamesinin sorumluluğundan ayrılmasının ayrı işlevsel süreçleri belirtmesi veya geliştirilmiş bir yazılım paketinde sistem yöneticisi için 'normal' kullanıcılara sağlana işlevsellikten ayrı olarak paketin parametrelerini idame ettirme işlevselliğinin olması.)
K-07	**Giriş (G)** a) Tetikleyici bir Giriş'teki veri grubu basitçe yazılıma 'bir Y olayı oldu' bilgisini veren yalnızca bir veri özniteliğinden oluşabilir. Çok sıkça, özellikle de iş uygulamaları yazılımlarında, tetikleyici Giriş yazılıma 'bir Y olayı oldu ve bu veriler de bu olaya ilişkin' bilgisini veren birkaç veri özniteliğinden oluşur. b) Tetikleyici olaylar olan saat-vuruşları her zaman ölçülecek yazılımın dışındadır. Bunun için, örneğin, her 3 saniyede bir olan saat-vuruşu olayı bir veri grubunu hareket ettiren bir Giriş ile ilişkilendirilir. Burada tetikleyici olayın donanım veya uygulama sınırı dışındaki başka bir yazılım parçası tarafından periyodik olarak oluşturuyor olması bir fark yaratmadığına dikkat edilmelidir. c) Bir işlevsel süreç gerektirmediği sürece, zaman bilgisinin sistem saatinden alınması bir Giriş'e sebep olduğu kabul edilmez. d) Belirli bir olayın oluşumu bir ilgi nesnesine ilişkin 'n' tane veri özniteliğinden oluşan bir veri grubunun Giriş'ini tetikliyorsa ve İKG, aynı olayın diğer oluşumlarının bu ilgi nesnesine ilişkin 'n' özniteliğinin bir alt kümesinden oluşan bir veri grubunun Giriş'ini tetiklemeye müsaade ediyorsa; bu durumda 'n' tane veri özniteliğinden oluşan tek bir Giriş belirlenir.
K-08	**Çıkış (Ç)** a) Kullanıcı verisi olmadan yazılım tarafından oluşturulan her türlü mesaj ve çıktı (örneğin hata mesajları) bir ilgi nesnesine ilişkin tek bir özniteliğe ait değerler olarak kabul edilir ('hata bildirimi' olarak isimlendirilebilir). Bunun

NO	KURAL TANIMI
	için, İKG'de ihtiyaç duyulan ve her işlevsel süreçte bulunan bütün bu mesaj oluşumları tek bir Çıkış olarak belirlenir. b) Bir işlevsel süreçteki bir Çıkış, bir ilgi nesnesine ilişkin 'n' tane veri özniteliğinden oluşan bir veri grubunu hareket ettiriyorsa ve İKG, işlevsel sürecin bu ilgi nesnesine ilişkin 'n' özniteliğinin bir alt kümesinden oluşan bir veri grubunu hareket ettiren bir Çıkış'ın oluşumuna müsaade ediyorsa; bu durumda 'n' tane veri özniteliğinden oluşan tek bir Çıkış belirlenir.
K-9	**Veri hareketi benzersizliği ve olası istisnalar** a) İşlevsel Kullanıcı Gereksinimleri aksini belirtmediği sürece, bir işlevsel sürece giriş olması gereken herhangi bir ilgi nesnesini tanımlayan tüm veri öznitelikleri ve ilişkili tüm veri işlemeleri tek bir Giriş (tipi) olarak belirlenmeli ve sayılmalıdır. (Not: Bir işlevsel süreç, elbette ki, her biri farklı bir ilgi nesnesini (tipini) tanımlayan bir veri grubunu hareket ettiren, birden çok Giriş tipini ele alması/içermesi gerekebilir. Aynı eşdeğer kural herhangi bir işlevsel süreçteki herhangi bir Okuma, Yazma veya Çıkış veri hareketi için geçerlidir. b) Birden çok Giriş için bir İşlevsel Kullanıcı Gereksinimi var ise, bir işlevsel süreçte her biri aynı ilgi nesnesini (tipini) tanımlayan veri gruplarını hareket ettiren birden çok Giriş veri hareketi (tipi) belirlenebilir ve sayılabilir. Benzer olarak, birden çok Giriş için bir İşlevsel Kullanıcı Gereksinimi var ise, aynı işlevsel süreçte aynı veri grubunu (tipini) hareket ettiren fakat her biri farklı veri işlemesi (tipi) ile ilişkili birden çok Giriş veri hareketi (tipi) belirlenebilir ve sayılabilir. Bu tür İKG, birden çok Girişin bir işlevsel süreçte (aynı ilgi nesnesini tanımlayan) farklı veri gruplarını giren farklı işlevsel kullanıcıların varlığından kaynaklandığı durumlarda ortaya çıkabilir. Aynı eşdeğer kural herhangi bir işlevsel süreçteki herhangi bir Okuma, Yazma veya Çıkış veri hareketi için geçerlidir. c) Tekrarlanan veri hareketi tipi oluşumları (yani aynı veri işlemelerine sahip aynı veri grubunun hareketi) herhangi bir işlevsel süreçte birden çok kez belirlenmemeli ve sayılmamalıdır. d) Eğer, bir işlevsel süreçteki veri hareketinin çoklu oluşumu, ilgili veri işlemeleri bazında, veri grubuna ait farklı veri öznitelikleri değerlerinin hareket ettirilmesinin farklı işleme yollarına neden olması dolayısıyla değişiyorsa, veri hareketi tipi o işlevsel süreçte birden çok kez belirlenmemeli ve sayılmamalıdır.
K-10	**Bir işlevsel sürecin işlevsel kullanıcıdan veri temin etmesi durumunda** a) Aşağıdaki dört durumda olduğu üzere, bir işlevsel süreç işlevsel kullanıcıya hangi veriyi göndereceğini bildirmesi gerekmiyor ise, o işlevsel süreç işlevsel kullanıcıdan bir veri grubunu bir Giriş veri hareketi ile edinmelidir: • bir işlevsel kullanıcı, işlevsel süreci başlatan bir tetikleyici Giriş gönderdiğinde; • bir işlevsel süreç, tetikleyici Giriş almasının ardından, duraksayıp işlevsel kullanıcıdan ilave Giriş beklediğinde (insan işlevsel kullanıcıların iş uygulamaları yazılımlarına veri girişlerinde oluşabilir); • bir işlevsel süreç, başlamasının ardından, işlevsel kullanıcıdan 'veriniz varsa şimdi iletin' mesajı ilettiğinde ve işlevsel kullanıcı mesajını ilettiğinde; • bir işlevsel süreç, başlamasının ardından, işlevsel kullanıcının durumunu incelediğinde ve kendisine gereken veriyi aldığında. Son iki durumda (genellikle gerçek-zamanı 'tarama' (polling) yazılımlarında

NO	KURAL TANIMI
	karşılaşılır) kural gereği gereken verinin alınmasında işlevsel süreçten bir Çıkış tanımlanmamalıdır. İşlevsel sürecin işlevsel kullanıcıya sadece bir ileti mesajı göndermesi gerekir ve bu ileti mesajının işlevselliği Girişin bir parçası olarak değerlendirilir. İşlevsel süreç nasıl bir veri beklediğini bilir. Bu durumda sadece bir Giriş gereklidir. b) "Bir işlevsel sürecin, bir işlevsel kullanıcının servisine ihtiyacı olduğu (örneğin veri edinme amacıyla) ve *işlevsel kullanıcının ne göndereceği hususunda bilgilendirilmesi gerektiği durumda* (genellikle işlevsel kullanıcının ölçülen yazılımın kapsamı dışında başka bir yazılım olduğu durumda), bir çift Giriş/Çıkış veri hareketi belirlenmelidir. Çıkış, özel veri isteğini; Giriş, geri döndürülen veriyi içerir.
K-11	**İş uygulamaları alanında kontrol komutları** İş uygulamaları alanında 'kontrol komutları', bir ilgi objesi ile ilgili herhangi bir veri hareketi içermediğinden göz ardı edilmelidir.
K-12	**Ölçüm sonuçlarının toplanması** a) Herhangi bir işlevsel süreç için, ayrı veri hareketlerinin işlevsel büyüklükleri aritmetik olarak eklenerek CİP birimi olarak toplanmalıdır. Büyüklük (işlevsel süreç) = Σ büyüklük(Girişler$_i$) + Σ büyüklük(Çıkışlar$_i$) + Σ büyüklük(Okumalar$_i$) + Σ büyüklük(Yazmalar$_i$) b) Herhangi bir işlevsel süreç için, İşlevsel Kullanıcı Gereksinimlerindeki değişimlerin işlevsel büyüklüğünü CİP birimi olarak vermek için işlevsel sürecin eklenen, değiştirilen veya silinen veri hareketleri aşağıdaki formüle göre toplanmalıdır. Büyüklük (Değişim(işlevsel süreç $_i$)) = Σ büyüklük (eklenen veri hareketleri $_i$) + Σ size (değiştirilen veri hareketleri $_i$) + Σ size (silinen veri hareketleri $_i$) İşlevsel büyüklüğün toplanması hakkında daha fazla bilgi için bakınız kısım 4.3.2. Değiştirilen yazılımın büyüklüğünün ölçümü için bakınız kısım 4.4. c) Tanımlanmış bir kapsam içerisinde, bir yazılım parçasının büyüklüğünü bulmak için yazılım parçasının işlevsel süreç büyüklükleri, aşağıdaki e) ve f) kurallarına göre toplanmalıdır. d) Tanımlanmış kapsam içerisinde, bir yazılım parçasındaki herhangi değişimin büyüklüğü, o parçaya ait bütün işlevsel süreçlerdeki değişimlerin aşağıdaki e) ve f) kurallarına göre toplaması ile elde edilir. e) Katmanlar arasındaki yazılım parçasının büyüklüğü veya yazılım parçasındaki değişimin büyüklüğü, sadece İKG'leri aynı işlevsel süreç tanesellik (granularity) seviyesinde ölçüldü iseler toplanabilirler. f) Herhangi bir katmandaki veya farklı katmanlardaki yazılım parçasının büyüklükleri ve/veya yazılım parçasındaki değişimlerin büyüklükleri sadece ölçümün amacı doğrultusunda toplanması anlamlı ise toplanabilir. g) Bileşenler arası veri hareketlerinin toplam büyüklük ölçüsüne katkısı ortadan kaldırılmadıkça bir yazılım parçasının boyutu, onun parçalarını toplayarak elde edilmez (parçanın birleşenlerine nasıl ayrıştırıldığından bağımsız olarak).

NO	KURAL TANIMI
	h) COSMIC yönteminin yerel bir uzantısı oluşturulursa, (örneğin standart yöntemin içerisinde ele alınmamış bir büyüklük tarafını ölçmek amacıyla), yerel uzantı ile ölçülmüş olan büyülük 5.1'de tanımlandığı gibi ayrı bir şekilde rapor edilmeli ve standart yöntem ile elde edilmiş olan CİP birimindeki büyüklüğe eklenmemelidir (daha detaylı bilgi için bakınız kısım 4.5.)
K-13	**Bir veri hareketinin değiştirilmesi** a) Bir veri hareketine ilişkin veri işlemelerindeki bir değişiklikten ve/veya hareket ettirilen veri grubundaki öznitelik sayısındaki veya tipindeki değişiklikten dolayı bir veri hareketi değiştirilmek durumunda ise; tek bir veri hareketinde yapılan değişikliklerin asıl sayısından bağımsız olarak değişiklik 1 CİP olarak ölçülmelidir. b) Bir veri grubunun değiştirilmesi gerekiyor ise, işlevselliği bu değişiklikten etkilenmemiş olan ve değiştirilmiş veri grubunu hareket ettiren veri grubunu taşıyan veri hareketleri, değiştirilmiş veri hareketleri olarak tanımlanmamalıdır. NOT 1: Özniteliğin değerindeki bir değişiklik, örneğin değerleri kodlama sistemi olan bir özniteliğin tekil kod değerinde yapılan bir değişiklik, özniteliğin tipinde yapılmış bir değişiklik değildir. NOT 2: Bir İşlevsel kullanıcıya göre bir ilgi nesnesi ile ilişkili olmayan ve giriş veya çıkış ekranlarında beliren herhangi bir verideki değişiklik, değişmiş bir CİP olarak değerlendirilmemelidir (bu gibi veriler için bakınız kısım 3.3.4).
K-14	**COSMIC ölçme etiketleme** Bir COSMIC ölçme sonucu "***x*** CİP (v.***y***) " olarak ifade edilir. Burada, • "**x**" işlevsel büyüklüğün sayısal değerini ifade eder, • "v.**y**" sayısal işlevsel büyüklük değeri "x"'i elde etmede kullanılan standart COSMIC yöntemi sürümünü ifade eder. NOT: Eğer ölçme için yerel yaklaştırma yöntemi kullanılmış, ancak ölçme bir standart COSMIC sürümü kurallarına göre yapılmışsa, yukarıdaki etiketlendirme geleneği kullanılır, ancak yaklaştırma yöntemi bir yere not edilir – bölüm 5.2'ye bakınız.
K-15	**COSMIC yerel genişletmeler etiketlendirmesi** Yerel genişletmeler kullanılarak elde edilen bir COSMIC ölçme sonucu şu şekilde ifade edilir: " ***x*** CİP (v. ***y***) + z Yerel İP", burada: • "**x**" standart COSMIC yöntemi sürüm v.y'ye göre yapılan her bir ölçme sonucunun toplanması ile elde edilen sayısal değeri ifade eder. • "v.**y**" sayısal işlevsel büyüklük değeri "x"'i elde etmede kullanılan standart COSMIC yöntemi sürümünü ifade eder. • "z" standart COSMIC yönteminin yerel genişletmelerine göre yapılan her bir ölçme sonucunun toplanması ile elde edilen sayısal değeri ifade eder.
K-16	**COSMIC ölçme raporlama** 5.1'deki gibi kaydedilmiş gerçek ölçümlere ek olarak, her ölçümle ilgili aşağıdaki öznitelikler de kayıt edilmelidir. a) Ölçülen yazılım bileşeninin tanımlanması (adı, sürüm No veya konfigürasyon No). b) Ölçme için kullanılan İKG'nin belirlenmesi için kullanılan bilgi kaynaklar.

NO	KURAL TANIMI
	c) Yazılımın alanı. d) Ölçmenin amacını içeren ifade. e) Ölçüm kapsamının tanımı ve varsa diğer ilişkili ölçümlerin toplam kapsamıyla olan ilişkisi. (Bölüm 2.2'de verilen genel kapsam kategorilerini kullanınız) f) Yazılımın İşlevsel Kullanıcıları. g) İKG'nin tanesellik seviyesi ve yazılımın ayrıştırma seviyesi. h) Proje yaşam döngüsü içinde ölçümün gerçekleştirildiği noktanın tanımı. (Ölçümün tamamlanmamış İKG'ler üzerinden gerçekleştirilen bir kestirim mi olduğu yoksa teslim edilen işlevsellik üzerinden gerçekleştirilen bir ölçüm mü olduğu belirtilmelidir.) i) Ölçümün hedeflenen veya varsayılan hata payı. j) Standart COSMIC yöntemi kullanılmışsa veya yönteme herhangi bir yereli yaklaşım getirilmişse ve/veya yönteme yerel genişletmeler uygulanmışsa bunların belirtilmesi. (Bölüm 4.5'e bakınız). Bölüm 5.1 ve 5.2'de verilen etiketleme standartlarını kullanınız. k) Ölçümün geliştirilen işlevsellik mi teslim edilen işlevsellik için mi olduğunun belirtilmesi. ("Geliştirilen işlevsellik" yeni geliştirilen yazlımın büyüklüğünü ifade eder; "teslim edilen işlevsellik" geliştirilen işlevselliği içermekle beraber, yeni geliştirme dışındaki yollarla elde edilen işlevselliği de kapsar. (örn: her türlü tekrar kullanım, var olan işlevselliği ekleyip çıkarmak için kullanılan parametreler vb.) l) Ölçümün yeni oluşturulan işlevsellik için mi yoksa bir bakım/değişiklik etkinliği sonucunda mı gerçekleştirildiğinin belirtilmesi. (Eklenen, değiştirilen ve çıkartılan işlevselliğin toplamı. Bkz. 4.4) m) Varsa, ölçümün yapıldığı katmanlara ait mimarinin tanımı. n) Varsa, toplam büyüklüğü oluşturmak için büyüklükleri toplanan ana bileşenlerin sayısı. o) Toplam ölçüm kapsamı içindeki her kapsam için, Ek-A'da belirtildiği gibi bir ölçüm matrisi. p) Ölçümü yapan kişinin adı ve sahip olduğu COSMIC sertifikasyon dereceleri.

EK D – COSMIC YÖNTEMİNİN SÜRÜMLERİNİN TARİHÇESİ.

Bu ek, COSMIC işlevsel büyüklük yönteminin versiyon 2.2'den 3.0 versiyonu türetilme sürecinde yapılan belli başlı değişiklikleri içermektedir. Yöntemin 2.2 versiyonu, 'Ölçüm Kılavuzu v2.2' de (MM v2.2 olarak kısaltılacaktır) tamamen açıklanmıştır. Fakat v3.0 dokümantasyonu şu anda sadece birinin ismi 'Ölçüm Kılavuzu' olmak üzere dört adet doküman olarak dağıtılmaktadır.

Bu ekin amacı MM v2.2 ile ilgili bilgi sahibi olan okuyucunun yapılan değişiklikleri ve gerekli yerlerde de gerekçelerini izleyebilmesini sağlamaktır. (Versiyon 2.2'nin versiyon 2.1'den türetilmesi esnasında yapılan değişiklikler için lütfen Ölçüm Kılavuzu V2.2 Ek E'ye başvurunuz.)

Versiyon 2.2 den versiyon 3.0'e

Aşağıda iki 'Yöntem Güncelleme Bülteni' ne (ya da kısaltma olarak 'MUB') atıf yapılmıştır. MUB, COSMIC tarafından yöntem tanımının majör sürümleri arasında önerilen iyileştirmeleri içerecek şekilde yayımlanır. Sözü edilen iki MUB aşağıda verilmiştir:

- MUB 1 Mayıs 2003'te yayımlanan "Yazılım 'katman' tanım ve özelliklerine önerilen iyileştirmeler".
- MUB 2 Mart 2005'te yayımlanan "'İlgi Nesnesi' tanımına önerilen iyileştirmeler".

Yöntemi v2.2'den v3.0'a güncellerken 'ilkeler' ve 'kurallar'ın arasında olduğu gibi ve bütün örnekleri ilkeler ve kurallardan ayırırken kılavuzluğun iyileştirilmesi için çaba harcanmıştır. Bunun gibi değişiklikler ve birçok editörlük iyileştirmeleri aşağıda açıklanmamıştır.

V2.2 Ref	V3.0 Ref	Değişiklik
COSMIC Yöntem Yeniden Yapılandırması		
2.2, 2.7, 3.1, 3.2	1.5, Bölüm 2	'Ölçüm Strateji' aşaması şimdiki üç aşamalı yöntemin ilk aşaması olarak ayrıldı. Ölçüm Strateji aşaması, daha önce MM v2.2'de Eşleme Aşamasının bir parçası olarak düşünülen 'katmanlar', 'sınırlar' ve '(işlevsel) kullanıcılar' kavramlarının düşünülmesini de artık kapsamaktadır.
Ölçüm Kılavuzu Yeniden Yapılandırması		
		COSMIC yönteminin v3.0 versiyonu oluşturulurken MM v2.2 kullanım kolaylığı için dört dokümana bölünmüştür. • 'COSMIC Yöntemi v3.0: Dokümantasyona Genel Bakış ve Terimler Sözlüğü' (yeni) • 'COSMIC Yöntemi v3.0: Yönteme Genel Bakış' (MM v2.2 Bölüm 1 ve 2'den) • 'COSMIC Yöntemi v3.0: Ölçme Kılavuzu' (MM v2.2 Bölüm 3, 4 ve 5 ve Ekler'den) • 'COSMIC Yöntemi v3.0: İleri ve İlgili Başlıklar' (MM v2.2 Bölüm 6 ve 7'den) Geçmiş deneyimlere ve araştırma makalelerine referanslar MM'den çıkarılmıştır. Şu an www.gelog.etsmtl.ca/cosmic-ffp adresinden erişilebilir durumdadırlar. Bu dört dokümanda kalan referanslar dipnotlar'da

		verilmiştir.
Bölüm 4	Bölüm 4	Ölçme Aşaması üzerine yazılmış olan bölüm daha mantıklı bir sunuş için önemli ölçüde yeniden yapılandırılmıştır.
Ek D	--	'Yazılım katmanları hakkında daha fazla bilgi' isimli ek MUB 1 ile uyumsuz olduğu ve çok az değer kattığı için kaldırılmıştır. MUB 1'i hesaba katan değişikliklerle ilgili aşağıdaki notlara da bakınız.
İsimler ve terminoloji değişiklikleri		
Genel		'COSMIC-FFP' olan yöntem ismi 'COSMIC' olarak basitleştirilmiştir.
5.1	4.2	Ölçüm birimi ismi olan 'COSMIC işlevsel büyüklük birimi' ('Cfsu' olarak kısaltılır) 'COSMIC İşlev Puan' (CFP olarak kısaltılır) olarak değiştirilmiştir. Bu konuya ilişkin bir açıklama için bu kılavuzun (MM v3.0) önsözüne başvurunuz.
4.1	4.1.7	'Veri hareketi tekrardan arındırma' kuralı 'Veri hareketi tekilliği' kuralı olarak değiştirilmiştir.
Yeni, değiştirilen ve kaldırılan kavramlar		
2.7	2.3	'Soyutlama', 'bakış açısı', 'Ölçme Bakış açısı', 'Son kullanıcı Ölçme Bakış açısı' ve 'Geliştirici Ölçme Bakış açısı' kavramları kaldırılmıştır. Bunların yerini, daha genel bir kavram olan ölçülecek yazılım parçasının büyüklüğünün o yazılımın 'işlevsel kullanıcılarına' sunulan işlevselliğe bağlılığı kavramı almıştır. Bunlar ölçülecek yazılımın işlevsel kullanıcı gereksinimlerinden saptanabilir olmalıdır. Bu sebeple 'işlevsel kullanıcıların' tanımlanması hangi büyüklüğün ölçülmesi gerektiğinin tanımlanması veya var olan bir büyüklük ölçümünün yorumlanması için ön koşuldur. Bu değişikliğe ilişkin detaylı bir açıklama için bu kılavuzun (MM v3.0) önsözüne başvurunuz.
3.2	2.3	'Kullanıcı' (ISO/IEC 14143/1 de tanımlandığı haliyle) kavramının yerini daha kısıtlı bir kavram olan 'işlevsel kullanıcı' kavramı almıştır. Bu değişikliğe ilişkin detaylı bir açıklama için bu kılavuzun (MM v3.0) önsözüne başvurunuz. 2.3.2'ye işlevsel büyüklüğün İKG'den saptanan işlevsel kullanıcının tipi ile değiştiği iki örnek eklenmiştir. Bu örnekler, bir mobil telefon ve iş uygulama yazılım paketidir.
--	2.2.2	Ölçme kapsamındaki tartışmaya ölçülecek yazılımın 'Ayrıştırma seviyesi' kavramı ilave edilmiştir.
--	2.4	Ölçme stratejisindeki tartışmaya, ölçülecek yazılımın işlevsel kullanıcı gereksinimlerinin 'tanesellik ölçüsü seviyesi' kavramı ilave edilmiştir. Kapsamlı bir örnek verilmiştir. Bu değişikliğe ilişkin detaylı bir açıklama için bu kılavuzun (MM v3.0) önsözüne başvurunuz.
--	2.4.3	'İşlevsel süreç tanesellik seviyesi' tanımlanmış ve kurallar ve tavsiye verilmiştir.'
3.4	4.2	'Geçici', 'kısa' ve 'Süresiz' isimli üç kalıcılık seviyesini içeren 'veri grubu kalıcılığı' gereksiz görüldüğü için çıkarılmıştır. 'kalıcı hafıza' kavramı

		tanıtılmıştır.
4.1	--	'Veri hareketi tekrardan arındırma' kavramı gereksiz görüldüğü için çıkarılmıştır.
İyileştirilmiş ve işlenmiş tanımlar, ilkeler ve kurallar		
2.3	2.2	'İşlevsel Kullanıcı Gereksinimleri' tanımı, ISO/IEC 14143/1 in 2007 baskısına uygun hale getirilmek için değiştirilmiştir.
2.4.1	1.3	'Yazılım Bağlam Modeli' bir prensipler beyanı olarak genişletilmiş ve işlenmiştir.
2.4.2	1.4	'Genel Yazılım Modeli' bir prensipler beyanı olarak genişletilmiş ve işlenmiştir.
2.4, 3.1	2.2.3, 2.2.4, 3.1	'Katman' tanımı MUB1'i hesaba katacak şekilde güncellenmiştir. MM v2.2'deki kullanıcıların yazılım ile 'katmanlar' üzerinden etkileşimini gösteren 2.4.1.1, 2.4.1.2 ve 2.4.1.3 şekilleri, v3.0'da tipik katmanlı mimarileri gösteren 2.2.3.1 ve 2.2.3.2 resimleri ve işlevsel kullanıcıların yazılım ile mantıksal etkileşimini gösteren 3.1.1 ve 3.1.2 şekilleri ile değiştirilmiştir. Bu değişikliğin amacı, tipik katmanlı yazılım mimarisinin fiziksel görünüşü ile COSMIC modele göre ölçülecek olan yazılım parçası ile etkileşen işlevsel kullanıcının mantıksal görünüşü arasındaki ayırımı daha net yapmaktır.
--	3.2.4	'Eş bileşen' kavramı tanımlandı ve MUB 1 hesaba katılarak ilkeleri belirlendi.
2.5	4.2, 4.3	V2.2'de açıklanan Ölçme sürecinin 'Karakteristikleri' v3.0'da bir ilkeler ve kurallar kümesi olarak yeniden yazılmıştır.
2.6	2.4	MM v3.0 bölüm 2.4'te 'Tanesellik ölçüsü seviyesi' altında kısmen değinilen 'proje yaşamında erken büyüklük ölçümü: Ölçme ölçeklendirme' üzerine olan materyal, 'COSMIC Yöntemi v3.0: İleri ve İlgili Başlıklar' dokümanında ayrıntılı bir şekilde işlenmiştir.
2.7	2.2	Büyüklüğü ölçülecek bir çok ayrı yazılım parçasını içerebilecek bir ölçme çalışmasının 'Tüm kapsamı' ile özel bir büyüklük ölçümünün ayırımını yapabilmek için 'kapsam' kavramının tanımına bir not eklenmiştir.
3.2	4.1.8	'Ölçme bakış açıları'nın kılavuzdan çıkarılmasının ışığında, farklı 'ölçme bakış açıları' ile ve farklı katmanlar ile bir sınır üzerinden etkileşen kullanıcıları gösteren üç durum taşınıp yeniden yazılmıştır. Aşağıda 4.1.8'e başvurunuz (MM V3.0).
--	3.1	Genel Yazılım Modelinin ölçülen yazılıma uygulanmasına ilişkin bir ilke eklenmiştir.
3.3	3.2.1	'İşlevsel süreç' tanımı, bu tanımdaki 'aktör' kavramının yerine gelen 'işlevsel kullanıcı' kavramını hesaba katacak şekilde düzeltilmiştir.
3.3	3.2.1	'Tetikleyici olay' tanımı geliştirilmiştir.

--	3.2.1	İşlevsel kullanıcı, tetikleyici olay, tetikleyici giriş ve işlevsel süreç arasındaki ilişki şekil 3.1.1 ile açıklığa kavuşturulmuştur.
3.1	3.2.2	'İşlevsel süreç' için olan ilkeler ve kurallar gözden geçirilmiş bir kurallar kümesi ile birleştirilmiştir.
--	3.2.3, 3.2.4, 3.2.5	İşlevsel süreçler ile ilgili bir çok örnek ve nasıl ayırt edilecekleri ilave edilmiştir.
--	3.3.1	MUB 2 ile uyumlu olarak 'ilgi nesnesi' tanımı eklenmiştir.
3.4	3.3.3	İlgi nesnelerinin ve veri gruplarının saptanmasına ilişkin örnekler veri grupları için olan kurallardan ayrılmıştır. Varlık-ilişki veri analizi konvansiyonları üzerine olan bazı alana özel materyaller 'İş Uygulamaları İçin COSMIC Ölçüm Rehberi'v1.0' kılavuzuna taşınmıştır.
--	3.3.4	'veri hareketi adayı olmayan veri veya veri grupları' üzerine kılavuzluk yapılmıştır.
--	3.3.5	Tipik olarak bir gerçek zamanlı sistemde ne zaman 'işlevsel kullanıcı'yı verinin hareket ettirildiği ilgi nesnesinden ayırmanın gerekmediği durumlara ilişkin kılavuzluk yapılmıştır.
3.5	3.4	Veri özellikleri yöntemin zorunlu bir parçası olmadığı için 'veri özellikleri' hakkındaki tartışma azaltılmış ve basitleştirilmiştir.
4.1	4.1.1	'Veri hareketi' tanımı iyileştirilmiştir.
4.1	4.1.7	Yeni adı ile 'veri hareketi tekilliği ve olası istisnalar' konusu eklenen bir çok örnek ile netleştirilmiştir.
4.1	--	'Güncelle' işlevsel süreçlerindeki okuma ve yazma ile ilgili alana özel kurallar 'İş Uygulamaları İçin COSMIC Ölçüm Rehberi v1.0' kılavuzuna taşınmıştır.
4.1	4.1.8	'Sınırlardaki verilerin uygunluğuna ilişkin kurallar' (şu an silinmiş olan 'Geliştirici ölçme bakış açısı'nda MM v2.2'de uygulanmış olan) silindi. Fakat kavramlar MM v2.2'de bölüm 3.2'de verilen durumlar ile birleştirilerek 'İşlevsel süreç kalıcı hafızadan veya kalıcı hafızaya veriyi ne zaman taşır' isimli yeni bir bölüm oluşturulmuştur.
4.1	4.1.6	'Veri manipülasyonu' tanımlanmış ve 'veri hareketleri ile birleşmiş veri manipülasyonu' ile ilgili bir ilke eklenmiştir. Farklı tipteki veri hareketleri ile birleştirilmiş veri manipülasyonu üzerine yönlendirmeler genişletilmiştir.
4.1.1	4.1.2	Girişe ilişkin ile ve kurallar iyileştirilmiştir. 'Giriş talebi'ne ilişkin yeni bir ilke ilave edilmiştir.
4.1.2	4.1.3	'son kullanıcı ölçme bakış açısına' olan atıf dahil olmak üzere çıkışa ilişkin ile ve kurallar iyileştirilmiştir.
4.1.3	4.1.4	Okumaya ilişkin ilkelerde iyileştirme yapılmıştır. 'Okuma talebi' işlevselliğine ilişkin yeni bir ilke ilave edilmiştir.

4.1.4	4.1.5	Yazmaya ilişkin ilkelerde iyileştirme yapılmıştır.
--	4.1.9	'İşlevsel süreç işlevsel kullanıcıdan ne zaman veriye ihtiyaç duyar ' üzerine yeni kurallar eklenmiştir.
--	4.1.10	Sadece iş uygulamaları yazılım alanında geçerli olan 'kontrol komutu' kavramına ilişkin tanım ve kural ilave edilmiştir.
4.1.5	4.5	'Yönteme lokal uzantılar' tartışması genişletilmiştir.
4.3	4.3	'Ölçme sonuçlarının birleştirilmesi' ilkeleri 'kurallara' dönüştürüldü ve bir yazılım parçasının büyüklüğünün bileşenlerinin büyüklükleri toplanarak bulunmasını da kapsayacak şekilde genişletildi. MM v2.2 bu tür kurallara sadece 'geliştirici ölçme bakış açısı' bağlamı ile atıf yapıyordu. Bu kısıt kaldırıldı.
--	4.4	'Yazılım değişikliklerinin büyüklük ölçümü' üzerine yeni bir bölüm ve yeni kurallar eklendi.
5.1	5.1	Ölçme sonuçlarının etiketlenmesine dair kurallar, ölçüm biriminin 'Cfsu' dan 'CİP' ye değişmesini dikkate alacak şekilde değiştirildi.
5.2	5.2	'Ölçme raporlama' ya ilişkin kurallar listede daha fazla madde olacak şekilde genişletildi.
Ek B	Ek B	v3.0 İlkeleri güncellendi.
Ek C	Ek C	v3.0 Kurallar güncellendi.

Versiyon 3.0 den versiyon 3.0.1'e

Versiyon 3.0'den versiyon 3.0.1 türetilirken yapılan en önemli değişiklikler bazı tanım, ilke ve kuralların ve metnin bazı kısımlarının ifade tarzlarının iyileştirilmesidir. İstisnai bir hata dışındaki tüm iyileştirmeler açıklık amacıyla yapılmıştır. Güncellemelerin çoğu, 3.0.1 versiyonundan önceki üç adet yöntem güncelleme bülteninde yayınlanmıştır.

- MUB 3: Haziran 2008'de yayınlanan 'COSMIC Yöntemi v3.0 Ölçme Kılavuzundaki Şekil 4.1.8.1 (b) deki bir hatanın düzeltilmesi'
- MUB 4: Haziran 2008'de yayınlanan 'COSMIC Yöntemi v3.0 Ölçme Kılavuzundaki 'Giriş Talebi' işlevselliğine ilişkin ilke ve kuralların netleştirilmesi'
- MUB 5: Şubat 2009'da yayınlanan '(a) 'Ayrıştırma Seviyesi' ve 'Eş Bileşenler' tanımları ve (b) 'Eş Bileşenler' için olan c) ilkesine önerilen iyileştirmeler'

Ek olarak bazı editörlükle ilgili iyileştirmeler yapılmıştır. İlke olarak bu iyileştirmeler farklı bir yazı tipi kullanarak örneklerin ana metinden ayrılması ve daha fazla alt bölüm ilave edilmesi gibi şeylerdir.

V3.0.1 Ref	**Değişiklik**
2.2.3	'Ayrıştırma Seviyesi' tanımı daha açık hale getirildi (MUB 5).
2.2.4	'Katman' tanımındaki 'Yazılım Mimarisi', 'Mimari' terimi, yazılımın zaten

	bölümlenmiş olduğunu ima ettiği için 'Yazılım Sistemi' olarak değiştirildi. Kural c) kaldırıldı. Bu kural, ölçme ve katmanlar konusuna özel olmayıp yazılım tasarımındaki genel bir kuraldı. V3.0'deki Ek D atıfı yanlıştı.
2.2.5	'Eş' tanımı eklendi ve 'Eş Bileşen' tanımı daha açık hale getirildi.'Eş Bileşen' için olan c) ilkesi FSM'ye daha uygun olan ilke ile değiştirildi. Şekil 2.2.5.1, yazılım eş parçaları ve eş bileşenler arasındaki ilişkiyi daha açık hale getirmek için eklendi. (MUB 5).
2.3.2	Sınır konusundaki c) ilkesi daha açık hale getirilmek üzere değiştirildi. (MUB 5'in bir sonucu olarak)
2.4.3	Şekil 2.4.3.1'deki aşağıdaki iki kutuda sağ taraftaki köşede bulunan 'ödeme yöntemi', çek, kredi kartı vs. için daha uygun bir terim olması açısından 'Ödeme olanakları' olarak değiştirildi.
3.2.2	İşlevsel sürece ilişkin e) kuralı bazı sözcükler italik yapılarak kısıtlı hale getirildi. Mevcutta 'İşlevsel bir süreç, bir yazılım parçasının bir ve yalnızca bir katmanının ölçme kapsamında olmalıdır.' şeklindedir. Bu kısıt mevcutta 'İş Uygulamaları İçin COSMIC Ölçüm Rehberi' v1.1'inde zaten bulunmaktadır fakat aslında herhangi bir yazılım alanı için geçerlidir.
3.2.6	'Eş bileşenlerin işlevsel süreçleri' başlıklı bir bölüm eklendi. Bu metin çoğunlukla 'İş Uygulamaları İçin COSMIC Ölçüm Rehberi' v.1.1 versiyonundan alındı fakat aslında herhangi bir yazılım alanı için geçerlidir. Bu, 3.2.2'deki e) kuralının değiştirilmesi ile ilişkilidir.
4.1.2	Girişe ilişkin c) ilkesi 'Giriş Talebi' işlevselliğini daha iyi açıklamak üzere değiştirildi. (MUB 4).
4.1.7	'Veri Tekilliği ve Olası İstisnalar' üzerine olan bölümün giriş cümlesi a) kuralı ile tutarlı hale gelmesi ve denmek isteneni daha iyi ifade etmek için değiştirildi. Örnek 2 de daha açık hale getirmek için büyük ölçüde değiştirildi.
4.1.8	Şekil 4.1.8.1 (b), bir cihaz sürücü yazılımının donanım ile nasıl etkileştiği konusundaki bir hatayı düzeltmek için değiştirildi (MUB 3).
4.1.9	'İşlevsel süreç işlevsel kullanıcıdan ne zaman veri talep eder' konusuna ilişkin kurallar ve ilgili örnekler daha açık hale getirilmek amacıyla değiştirildi (MUB 4).

EK E - COSMIC DEĞİŞİKLİK TALEBİ VE YORUM PROSEDÜRÜ.

COSMIC Ölçme Uygulama Komitesi (MPC) geribildirim, yorum ve gerektiğinde COSMIC Ölçme Kılavuzuna yapılmak istenen değişiklik taleplerini almak konusunda oldukça heveslidir. Bu ek, COSMIC MPC ile iletişimin nasıl yapılacağını göstermektedir.

COSMIC MPC ile yapılacak olan tüm iletişimler e-posta ile ve aşağıdaki adres kullanılarak yapılmalıdır:

mpc-chair@cosmicon.com

Resmi Olmayan Genel Geribildirim ve Yorumlar.

Ölçme Kılavuzu ile ilgili olarak COSMIC yöntemini anlamakta veya uygulamakta karşılaşılan güçlükler, genel iyileştirme amaçlı tavsiyeler vb. gibi resmi olmayan yorumlar ve/veya geribildirimler yukarıda verilen e-posta adresine iletilmelidir.

Mesajlar kayıt altına alınacak ve genellikle iki hafta içerisinde alındığı bildirilecektir. MPC bu tür genel yorumlara ilişkin aksiyon alınacağını garanti edememektedir.

Resmi Değişiklik Talepleri.

Ölçme Kılavuzu okuyucusu metinde bir hata olduğuna inanıyorsa, netleştirilmeye muhtaç bir durum görüyorsa veya bazı metinlerin iyileştirilmesi gerektiğini düşünüyorsa resmi bir değişiklik talebi ('CR') gönderebilir.

Resmi CR'lar alındıktan sonra iki hafta içinde alındığına dair bilgi verilecektir. Her bir CR bir seri numara ile numaralandırılacak ve dünya çapında uzmanlar grubundan oluşan COSMIC MPC üyeleri arasında dolaşıma sokulacaktır. Normal incelemeleri en az bir aylık bir süreyi kapsar ve CR'ın kapsamına göre daha uzun da sürebilir.

İncelemenin sonucu olarak söz konus CR kabul edilebilir, reddedilebilir veya 'daha fazla tartışma için askıda' (Son duruma örnek olarak bir başka CR'a bağımlılık verilebilir) durumuna alınabilir ve sonuç CR'ı gönderene uygulanabilir olan en kısa sürede bildirilir.

Resmi bir CR sadece aşağıdaki bilgilerle dokümante edilirse kabul edilecektir.

- CR'ı gönderen kişinin isim, pozisyon ve kuruluşu;
- CR'ı gönderen kişinin kontak detayları;
- Gönderim tarihi;
- CR'ın amacını belirten genel ifade (Örneğin '.. metninin iyileştirilmesi gerekli');
- Değiştirilecek, silinecek veya yerini alacak gerçek metin (ya da oraya açık referans);
- Önerilen ilave veya yerini alacak metin;
- Değişikliğin neden gerektiğine dair tam açıklama;

CR gönderimi için kullanılacak bir form www.cosmicon.com sitesinde bulunmaktadır.

COSMIC MPC'nin CR inceleme sonucu ile ilgili olarak kararı ve eğer kabul edildiyse hangi Ölçme Kılavuzunda bu CR'ın uygulanacağı kesindir.

COSMIC yönteminin uygulanması ile ilgili sorular.

COSMIC MPC, COSMIC yönteminin kullanımı ve uygulaması ile ilgili sorulara yanıt verememektedir. Yöntem için araç desteği, eğitim ve danışmanlık sağlayan ticari kuruluşlar bulunmaktadır. Lütfen daha fazla detay için www.cosmicon.com web sitesine başvurunuz.