JAVA İŞ UYGULAMALARI İÇİN ÇALIŞMA ZAMANINDA COSMIC İŞLEVSEL BÜYÜKLÜK HESAPLANMASI

RUN-TIME MEASUREMENT OF COSMIC FUNCTIONAL SIZE FOR JAVA BUSINESS APPLICATIONS

AHMET ATA AKCA

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

Bilgisayar Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2013

Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Bu çalışma jürimiz tarafından **BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Başkan	:….. Prof. Dr. İlyas Çiçekli
Üye (Danışman)	:….. Yrd. Doç. Dr. Ayça Tarhan
Üye (Eş Danışman)	: ….. Prof. Dr. Hayri Sever
Üye	:….. Yrd. Doc. Dr. İbrahim Aykut Erdem
Üye	:….. Yrd. Doç. Dr. Ergin Soysal
Üye	:….. Öğr. Gör. Dr. Mehmet Erkut Erdem

ONAY

Bu tez, .../.../2013 tarihinde Enstitü Yönetim Kurulunca kabul edilmiştir.

Prof. Dr. Fatma Sevin DÜZ
Fen Bilimleri Enstitüsü Müdürü

Aileme…

JAVA İŞ UYGULAMALARI İÇİN ÇALIŞMA ZAMANINDA COSMIC İŞLEVSEL BÜYÜKLÜK HESAPLANMASI

Ahmet Ata Akca

ÖZ

Yazılımların işlevsel büyüklüklerinin ölçülmesi, yazılım projelerinin yönetiminde önemi giderek artan bir konudur. Yazılım büyüklüklerinin, geliştirme sürecinin ilk evrelerinden, geliştirme sonuna kadar her adımda ölçülebilir olması, artan yazılım büyüklükleri sebebiyle giderek karmaşıklaşan proje yönetimi süreci için büyük önem taşımaktadır. İşlevsel büyüklük ölçme manüel olarak yapıldığında oldukça zaman alıcı ve maliyetli olmaktadır. Bu noktada ölçme işleminin otomatikleştirilmesi konusu ön plana çıkmıştır.

Bu tez kapsamında, üç katmanlı mimariye sahip Java iş uygulamalarının kullanıcı arayüzü üzerinden tetiklenen işlevsel süreçlerinin, geliştirdiğimiz “Measurement Kütüphanesi” kullanılarak keşfedilmesi ve bu işlevsel süreçler içinde gerçekleşen veri hareketlerinin izlenmesi yolu ile yazılımın COSMIC işlevsel büyüklüğünün, çalışma zamanında otomatik olarak ölçülmesi amaçlanmıştır. Kütüphane’nin kullanılabilmesi için, hazırlanan “Measurement Kütüphanesi ile Ölçme Kılavuzu” belgesinde belirtilen kod ekleme işlemlerinin yapılması gerekmektedir. Measurement Kütüphanesi kullanılarak gerçekleştirilen ön uygulamada kütüphane, basit bir öğrenci kayıt sistemi içinden (“import” edilerek) kullanılmış ve uygulamanın büyüklüğü, arayüz üzerindeki her işlevin manuel olarak bir kez tetiklenmesi sonrasında otomatik olarak ölçülmüştür. Otomatik ölçme sonuçlarının doğruluğunun karşılaştırılabilmesi amacıyla aynı uygulama için manuel ölçüm gerçekleştirilmiştir. Otomatik ölçülen ve manuel olarak hesaplanan COSMIC işlevsel büyüklüklerinin %92 oranında yakınsadığı görülmüştür. Alınan bu sonuçlar Measurement Kütüphanesi’nin doğru tasarlandığını ve uygulanabilir olduğunu göstermiştir. Daha sonraki aşamada, çalışmalarımız, Measurement Kütüphanesi kullanılarak yapılan ölçme işleminin manuel yapılan ölçme işlemiyle kıyaslandığında maliyet-etkin olup olmadığının belirlenmesi üzerine yoğunlaşmıştır. Yapılan üç farklı durum çalışması ile otomatik ölçme işleminin maliyet-etkinlik değerlendirmesi yapılmıştır. Elde edilen sonuçlar kütüphanenin yazılım geliştirme süreci tamamlandıktan sonra yazılıma entegre edildiği durumda, ekleme yapan kişinin yazılıma aşina olmadığı durum için maliyet-etkin olarak kullanılamadığını; kütüphanenin yazılım geliştirme sürecine en baştan entegre

edilmesi ile yapılan otomatik ölçümde ise manuel yapılan ölçme işlemine göre %500'lere varan oranda maliyet kazancı sağladığını göstermiştir.

Geliştirilen "Measurement Kütüphanesi"nin, yazılımların işlevsel büyüklüklerinin otomatik olarak ölçülmesi konusunda izlediği yöntem bakımından bir ilk olması sebebiyle, bu alanda yapılacak diğer çalışmalar için temel oluşturması beklenmektedir. Gelecek çalışmalarda kütüphane geliştirilerek daha geniş kapsamda kullanılabilir. Kütüphanenin yazılım geliştirme süreci tamamlandıktan sonra yazılıma entegre edildiği durumlar için, kütüphaneyi kullanmanın gerektirdiği kod ekleme ve veri hareketi tetikleme işlemlerinin de otomatikleştirilerek maliyetin düşürülmesi hedeflenmektedir.

Anahtar Sözcükler: İşlevsel büyüklük ölçümü, COSMIC yöntemi, çalışma zamanında ölçme, otomatik büyüklük ölçme, Java iş uygulamaları

RUN-TIME MEASUREMENT OF COSMIC FUNCTIONAL SIZE FOR JAVA BUSINESS APPLICATIONS

Ahmet Ata Akca

ABSTRACT

The issue of “Functional Size Measurement” is increasing in importance for the software project management. For the project management process, which is gradually complexifying because of the growing sizes of software, it is critical that the size of software should be measurable in every phase ranging from the first phases of development to the end of it. Considering the fact that it is considerably time-consuming and costly when functional size measurement is made manually; automating the process of measurement came to the fore.

In this study, it is aimed that the runtime measurement of COSMIC functional size shall be carried out automatically by ensuring functional processes, which are triggered via user interface of a three tier Java business application, to be discovered by using the Measurement Library that we developed and by monitoring the data movements which occurs during these functional processes. It is necessary to perform the code addition process that is specified in the “Measurement Library Manual” which is prepared to assist the user with the operation of the Library. The Library has been imported from a simple student registration system, and the application size has been measured automatically following the manual triggering of all functions which are operated on GUI at least once. In order to verify the results of the automatic measurement, the measurement was made manually once more. It has been found that the COSMIC functional sizes measured automatically and calculated manually are 92% approximate. Results of the measurement show us that the “Measurement Library” is well designed and applicable. Subsequently, the costs of automatic and manual measurements are compared and it is focused on whether automatic measurement is cost effective or not. In order to decide on this, cost effectiveness analysis of the automatic measurement is carried out with three different case studies. Although the Measurement Library is not cost effective when library is integrated after the development phase and/or user is not familiar with the library, it can decrease costs up to %500 compared to the manual measurement processes when it is integrated during early development phases.

Because of the fact that the “Measurement Library” is the first in its field in terms of the method used for the functional size measurement of software automatically, it is expected that it may provide a basis for further studies in this field. In future studies, scope of the Library may be improved. Also, the costs due to the integration of the Library after the development phase is planned to be decreased by also automating code additions and the trigeering of data movements.

Keywords: Functional Size Measurement, COSMIC method, run-time measurement, automatic size measurement, Java business applications

TEŞEKKÜR

Tez konusunun belirlenmesini sağlayan, tez çalışmasını yönlendiren, tez metnini içerik ve biçim bakımından inceleyerek şekillendiren Sayın Yrd. Doç. Dr Ayça Tarhan'a,

Tez metnini inceleyerek biçim ve içerik bakımından son halini almasına yardımcı olan Sayın Prof. Dr. Hayri Sever'e,

Tez metnini inceleyerek biçim ve içerik bakımından son halini almasına yardımcı olan Sayın Prof. Dr. İlyas Çiçekli'ye,

Tez metnini inceleyerek biçim ve içerik bakımından son halini almasına yardımcı olan Sayın Yrd. Doç. Dr. İbrahim Aykut Erdem'e

Tez metnini inceleyerek biçim ve içerik bakımından son halini almasına yardımcı olan Sayın Yrd. Doç. Dr. Ergin Soysal'a,

Tez metnini inceleyerek biçim ve içerik bakımından son halini almasına yardımcı olan Sayın Dr. Mehmet Erkut Erdem'e,

ve

Hiçbir zaman desteklerini esirgemeyen değerli aileme ve arkadaşlarıma teşekkür ederim.

İÇİNDEKİLER DİZİNİ

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

3GL	Third-generation programming language
AC	Actual Cost
AS	Araştırma Sorusu
BAC	Budget At Completion
CASE	Computer-Aided Software Engineering
CFP	Cosmic İşlevsel Büyüklük Birimi
CFSU	COSMIC Functional Size Unit
COSMIC	Common Software Measurement International Consortium
CPI	Cost Performance Index
CV	Cost Variance
DB	Database
DET	Data Element Type
E	Entry
EI	External Inputs
EIF	External Interface Files
ELOC	Effective Lines of Code
EO	External Outputs
EQ	External Inquiry
EV	Earned Value
EVM	Earned Value Management
FiSMA	Finnish Software Measurement Association
FFP	Full Function Point
FPA	Function Point Analysis
FSM	Functional Size Measurement
FUR	Functional User Requirements
GKA	Grafik Kullanıcı Arayüzü

GUI	Graphical User Interface
ICASE	Integrated Computer-Aided Software Engineering
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFPUG FPA	International Function Point Users Group Function Point Analysis
ILF	Internal Logical Files
ISO	International Organization for Standardization
KSS	Kod Satır Sayısı
LoC	Line of Code
MIS	Management Information System
NATO	North Atlantic Treaty Organization
NASA	National Aeronautics and Space Administration
NCLOC	Non Commented Lines of Code
NESMA	Netherlands Software Metrics Association
NOP	New Object Points
OVT	Öğrenci Veri Tabanı
PV	Planned Value
R	Read
RET	Record Element Type
RRRT	Rational Rose RealTime
SFSU	Scenario Functional Size Unit
SPI	Schedule Performance Index
SV	Schedule Variance
UFP	Unadjusted Funtion Points
UFSU	Use case Functional Size Unit
UKSMA	United Kingdom Software Metrics Association
UML	Unified Modeling Language

VT	Veri Tabanı
VTYS	Veri Tabanı Yönetim Sistemi
W	Write
WBS	Work Breakdown Structure
X	Exit
XML	Extensible Markup Language

1. GİRİŞ

1.1. Problem Tanımı

Ölçme, günlük hayat ile iç içe geçmiş, tüm bilimsel ve mühendislik disiplinleri için olmazsa olmaz bir kavramdır [1]. Herhangi bir konuda yapılan bir çalışmada değerlendirme yapabilmek için, ölçme işlemine ihtiyaç duyarız. İnşası yapılan bir evin büyüklüğü, içinde bulunan ortamın sıcaklığı, herhangi bir proje için harcanan zaman, ancak ölçme işlemi sayesinde belirlenebilir. Elde edilen sonuçlar değerlendirilerek sonraki çalışmalar için kaynak oluşturulabilir, içinde bulunan durumun analizi yapılabilir ve önceki çalışmalar için değerlendirmede bulunulabilir. Özellikle kapsamı içerisinde üretim planlama, izleme ve kontrol çalışmaları bulunan, karar alma nitelikli, fayda/maliyet analizi gerektiren ve deneyimleyerek öğrenme amaçlı çalışmalar, ölçme işleminden bağımsız düşünülemez [1].

Her ne kadar yarım asırlık tarihe sahip, nispeten yeni yazılım mühendisliği disiplininde yakın zamana kadar geri planda kalsa da ölçme eylemi, diğer mühendislik disiplinlerinde olduğu gibi bu disiplin için de hayati niteliktedir. Geliştirilen her üç yazılım projesinden ancak biri, hedeflerini karşılayarak ve başarıyla sonuçlanmaktadır [2]. Bir mühendislik disiplini ancak ölçme araçları kadar olgundur [3] ve yazılım ölçmenin mevcut durumu, yazılım mühendisliğinin bir disiplin olarak bulunduğu yeri bu açıdan ortaya koymaktadır. Yazılım süreçlerinin yönetimi, geliştirilen yazılım üzerinde iyileştirmelerin yapılması, yazılım geliştirme sürecinde harcanan işgücü ve katlanılan maliyet değerlendirmelerinin yapılması, ölçme sayesinde gerçekleştirilmektedir. Yazılım mühendisliğinde ölçme, yazılım süreçlerinin ve ürünlerinin anlaşılması, kontrol edilmesi ve mevcut başarımlarının izlenerek iyileştirilmesi etkinlikleridir [4].

Yazılımların büyüklüklerinin ölçülmesi, yazılım mühendisliği disiplini için önemi hızla artan bir konudur. Büyüklüğün yazılım geliştirmenin ilk adımlarından itibaren ölçülebilir olması, ölçme işlemini artan yazılım boyutları nedeniyle kritik önem kazanan yazılım proje yönetimi sürecinin en önemli bileşenlerinden biri haline getirmiştir. Erken evrelerde ölçme, proje planlamasının çok daha etkin gerçekleştirilebilmesini; bu sayede projenin belirlenen takvim içerisinde, hangi kaynakların kullanımıyla ve ne kadar maliyetle gerçekleştirilebileceğinin önceden belirlenebilmesini sağlar. Geliştirme sırasında yapılan ölçümler, planlanan yazılım

büyüklükleri ile karşılaştırılarak projenin gelişiminin yorumlanmasında kullanılabilir [5][6]. Sonraki projelerin planlamasında kullanılmak üzere, proje ekibine ait geçmiş bilgisi (“historical data”) kapsamında kaydedilerek saklanabilir. Ayrıca, yazılım hataları büyüklüğe göre normalize edilerek farklı yazılımların kaliteleri karşılaştırılabilir. Bütün bunlar proje yönetimi süreci için yazılım büyüklüklerinin kritik öneme sahip olduğunu açıkça ortaya koymaktadır.

Bir yazılım ürününün büyüklüğü dört boyutta ölçülmektedir [3]: Uzunluk, işlevsellik, karmaşıklık ve tekrar kullanma. Uzunluk boyutunda Kod Satır Sayısı metriği, en çok bilinen ve yaygın olarak kullanılandır. Kod satır sayısı metriği ortaya çıktığında, geniş kullanım alanı bulmuştur. Ancak gelişen teknolojilerle birlikte ortaya çıkan çok sayıda programlama yaklaşımı ile aynı işlevsellikler için gereken kod satır sayısı, oldukça değişken bir hale gelmiştir. Kod satır sayısı metriği uygulamanın geliştirildiği programlama diline ve platforma olan bağımlılığı sebebiyle farklı programlama dilleri ile geliştirilen yazılımlar için kıyaslanabilir bir sonuç üretmekten uzaktır.

İşlevsellik boyutunda İşlev Puan metriği, özellikle son yıllarda ön plana çıkmaktadır. İşlev Puan metriği bir yazılım ürününün işlevsel büyüklüğünü, yazılımın kullanıcılarına sunduğu işlevselliğin miktarı olarak ölçmeyi hedefler. Bu ölçümde ana fikir; yazılımları, geliştirilen dil ve platformdan bağımsız olarak, yalnızca sağladıkları işlevsellik üzerinden ölçmektir. Bu fikir ilk olarak 1979 tarihinde Allan Albrecht tarafından ortaya atılmıştır [5] ve zaman içerisinde kendi içinde farklılaşarak gelişmiştir. Halihazırda ISO tarafından onaylanmış beş farklı işlevsel büyüklük ölçme metodu vardır. Bu metotlardan kurumlarda ve akademik çalışmalarda en yoğun kullanılan COSMIC [7] ve IFPUG FPA [8] yöntemleridir. Birbirlerinden bazı noktalarda farklılık gösteren bu yöntemlerin birbirlerine karşı üstünlükleri yapılan çalışmalarla belirlenmeye çalışılmıştır [9][10][11].

Yazılım büyüklüklerinin manuel olarak ölçülmesi maliyetli bir işlem olduğundan dolayı akademik çevrelerce ilgi, bu işlemin otomatikleştirilmesi üzerine odaklanmıştır. Bu alanda yapılan birçok çalışma olmasına karşın [12][13][14][15][16] tam anlamıyla kabul görmüş bir yöntem bulunmamaktadır.

Bu tez kapsamında, yukarıda belirtilen problem tanımından hareketle Java iş uygulamaları için işlevsel yazılım büyüklüğünün; COSMIC [7] metodunun seçilen prensipleri çerçevesinde, çalışma zamanında otomatik olarak ölçülmesi

amaçlanmıştır. Bu amaçla yapılan çalışmaların ayrıntısına izleyen bölümde değinilmiştir.

1.2. Tez Çalışmasının Kapsamı

Tez kapsamında, üç katmanlı mimariye sahip Java iş uygulamalarının kullanıcı arayüzü üzerinden manuel olarak tetiklenen işlevsel süreçlerinin, geliştirdiğimiz "Measurement Kütüphanesi" kullanılarak keşfedilmesi ve bu işlevsel süreçler kapsamında gerçekleşen veri hareketlerinin izlenmesi yolu ile yazılımın COSMIC işlevsel büyüklüğünün, çalışma zamanında otomatik olarak ölçülmesi amaçlanmıştır. Çalışmada daha özel olarak aşağıdaki adımlar gerçekleştirilmiştir;

Anlama Evresi:

- ISO kapsamında kabul görmüş işlevsel büyüklük ölçümü yöntemlerinin incelenerek özelliklerinin ve farklılıklarının anlaşılması,
- Tez çalışması kapsamında kullanılan işlevsel büyüklük ölçme yöntemi olan COSMIC ölçme yöntemi ile ilgili ölçme prensipleri ve kuralları ile bu alanda yapılan çalışmaların ayrıntısıyla incelenmesi,
- Teze konu olan yazılım büyüklüklerinin ölçülmesinin otomatikleştirilmesi konusunda yapılmış çalışmaların incelenmesi ve öngördüğümüz kütüphanenin uygulanabilirliğinin değerlendirilmesi,
- Yapılan değerlendirme ışığında, COSMIC metodunun prensiplerinin gözden geçirilerek çalışmaya özel kapsamın seçilmesi ve gerçekleştirme için çerçevenin belirlenmesi.

Gerçekleme Evresi:

- Otomatik ölçme işleminin gerçekleştirilmesi amacıyla, uygulamalara entegre edilerek kullanılabilen "Measurement Kütüphanesi"nin belirlenen çerçeveye uygun olarak geliştirilmesi,
- Otomatik ölçme işleminin örnek bir uygulama üzerinde Measurement Kütüphanesi kullanılarak gerçekleştirilmesi ve ölçüm sonuçlarının aynı uygulama üzerinde gerçekleştirilen manuel ölçüm sonuçlarıyla kıyaslanması yoluyla başarımın analiz edilmesi,
- Elde edilen sonuçlar kullanılarak kütüphanenin uygulanabilirliğinin değerlendirilmesi,

Geçerleme Evresi:

- Measurement Kütüphanesi kullanılarak gerçekleştirilen otomatik ölçme işleminin maliyet-etkinlik değerlendirmesinin yapılabilmesi amacıyla durum çalışmalarının (Durum Çalışması A, Durum Çalışması B ve Durum Çalışması C) tasarlanması,
- Durum Çalışması A kapsamında, işlevsel büyüklük ölçme kavramı ve Measurement Kütüphanesi'nin kullanımı konularında bilgi sahibi olmayan bir yazılımcı tarafından (Yazılımcı 1) "İnternet Bankacılığı-1" isimli uygulamanın geliştirilmesi ve Measurement Kütüphanesi'nin bu geliştirme sürecine en başından itibaren entegre edilmesi ile işlevsel büyüklüğün otomatik olarak ölçülmesi ve harcanan sürenin kaydedilmesi,
- Durum Çalışması A kapsamında geliştirilen uygulama için, geliştirme sürecinin farklı evrelerinde manuel ölçüm yapılarak sonuçların ve harcanan sürenin kaydedilmesi,
- Durum Çalışması B kapsamında, işlevsel büyüklük kavramı ve özellikle COSMIC işlevsel büyüklük ölçme yöntemi konularında tecrübeli, Measurement Kütüphanesi'nin kullanımı konusunda bilgi sahibi bir yazılımcı tarafından (Yazılımcı 2) Durum Çalışması A kapsamında geliştirilen uygulama ile aynı işlevselliğe sahip "İnternet Bankacılığı-2" uygulamasının geliştirilmesi ve Measurement Kütüphanesi'nin bu geliştirme sürecine en başından itibaren entegre edilmesi ile işlevsel büyüklüğün otomatik olarak ölçülmesi ve harcanan sürenin kaydedilmesi,
- Durum Çalışması B kapsamında geliştirilen uygulama için, geliştirme sürecinin farklı evrelerinde manuel ölçüm yapılarak sonuçların ve harcanan sürenin kaydedilmesi,
- Durum Çalışması C kapsamında, Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü BİL342 kodlu ders kapsamında proje olarak verilen ve geliştirme süreci sona ermiş "Hastane Randevu Sistemi" uygulaması üzerine Yazılımcı 2 tarafından Measurement Kütüphanesi'nin entegre edilmesi ile otomatik işlevsel büyüklük ölçme işleminin gerçekleştirilmesi ve geçen sürenin kaydedilmesi,
- Durum Çalışması C kapsamında üzerinde çalışılan uygulama için manuel ölçüm yapılarak sonuçların ve harcanan sürenin kaydedilmesi,

- Durum çalışmaları ile elde edilen verilerin Measurement Kütüphanesi için değerlendirilerek sonuçların ortaya konması.

Çalışma kapsamında ilk olarak ISO tarafından kabul görmüş işlevsel büyüklük ölçüm yöntemleri ele alınmış ve ayrıntılı olarak incelenmiştir. İncelenen bu yöntemler aşağıda verilmiştir;

1. Mk II Function Point Analysis (FPA) [17]
2. International Function Point Users Group (IFPUG) FPA [18]
3. Common Software Measurement International Consortium (COSMIC) Functional Size Measurement Method [19]
4. The Netherlands Software Metrics Users Association (NESMA) Functional Size Measurement (FSM) [20]
5. The Finnish Software Metrics Association (FiSMA) FSM [21]

Tez çalışması kapsamında, ikinci nesil işlevsel büyüklük ölçme yöntemi olarak nitelenen COSMIC yöntemi kullanılmıştır. COSMIC yöntemi “COSMIC versiyon 3.0.1 Ölçme Kılavuzu” [7] ayrıntısıyla incelenerek anlaşılmıştır. Kodun statik analizi yerine çalışma zamanında kullanıcılar tarafından manuel olarak tetiklenen işlevsellik yoluyla otomatik ölçme amaçlandığından, yapacağımız çalışma için; hangi noktaların üzerinde durulması gerektiği ve hangilerinin göz ardı edilebileceği, hangilerinin yapacağımız ölçme kapsamında yer alabileceği ya da alamayacağı belirlenmiştir. Bu aşamada ayrıca, COSMIC tarafından hazırlanan “İş Uygulama Yazılımları için Ölçme Kılavuzu” [22] belgesi incelenmiş, içerisindeki temel düzeydeki basit uygulamalardan kendi uygulamamız için çıkarımlar yapılmıştır. Ardından COSMIC metodu kullanılarak yapılan çalışmalar [23][24][25][26] incelenerek, yöntemin nasıl uygulandığına dair izlenimler edinilmiştir.

Bir sonraki adımda, ölçme işleminin otomatikleştirilmesi konusunda yapılan çalışmalar incelenmiştir. Yapılan incelemelerde üzerinde durduğumuz, çalışma zamanında işlevsel süreç keşfi ve takibi yolu ile işlevsel büyüklük hesaplanması üzerine eğilen bir çalışma ile karşılaşılmamıştır. Hem yapılan çalışmalar hem de COSMIC ölçme yönteminin temelleri göz önüne alındığında, öngörülen kütüphanenin gerçekleştirilebileceği sonucuna ulaşılmıştır.

Gerçekleştirilebilirliğin değerlendirilmesinin ardından, çalışmaya konu olan kütüphanenin geliştirilmesine geçilmiştir. Önceki adımlarda toplanan bilgiler ve

yapılan değerlendirmeler ışığında, öncelikle geliştirimini yapacağımız kütüphanenin hangi kapsamda ve işlevsellikte olması gerektiği belirlenmiştir. Bunun için, COSMIC metodunun prensipleri gözden geçirilerek gerçekleştirme için çerçeve çizilmiştir. Uygulama adımları belirlenmiş ve “Measurement” kütüphanesi, işlevselliğinin ölçülmesi planlanan yazılımın koduna yapılması gereken müdahaleyi minimum tutma hedefiyle gerçekleştirilmiştir. Gerçekleştirme sonrasında hazırlanan kullanım kılavuzunda koda, hangi noktalarda ve ne şekilde müdahale edilmesi gerektiği ayrıntılı olarak anlatılmıştır.

Geliştirilen kütüphane, başarımını ölçmek amacıyla, geliştirilen temel düzeydeki bir öğrenci kayıt sistemi uygulaması üzerine entegre edilerek kullanılmıştır. Kütüphane kullanılarak gerçekleştirilen ölçme işlemi sonrasında elde edilen işlevsel büyüklük değeri ile manuel yapılan ölçme sonucu elde edilen değer karşılaştırılmıştır. Büyük oranda (%92) yakın çıkan sonuçlar, öngörülen yöntemin ve geliştirilen kütüphanenin, işlevsel büyüklük ölçümünün otomatikleştirilmesi amacıyla kullanılabileceğini göstermiştir.

Sonraki adımda ise Kütüphane’nin otomatik ölçme için kullanılması durumunda, manuel yapılan ölçme işlemine göre maliyet-etkin olup olmadığı gerçekleştirilen durum çalışmaları (Durum Çalışması A, Durum Çalışması B ve Durum Çalışması C) ile değerlendirilmiştir. Durum Çalışması A kapsamında işlevsel büyüklük ölçme kavramı ve Measurement Kütüphanesi’nin kullanımı konularında bilgi sahibi olmayan bir yazılımcı tarafından (Yazılımcı 1) “İnternet Bankacılığı-1” isimli uygulamanın geliştirimi sırasında Kütüphane, geliştirme sürecine en başından itibaren entegre edilmiş ve işlevsel büyüklük otomatik olarak ölçülmüştür. Durum Çalışması B kapsamında işlevsel büyüklük ölçme kavramı ve Measurement Kütüphanesi’nin kullanımı konularında tecrübe sahibi bir yazılımcı tarafından (Yazılımcı 2), işlevselliği Durum Çalışması A kapsamındaki uygulama ile aynı olan bir uygulama geliştirilmiş ve benzer şekilde kütüphane geliştirme sürecine en başından entegre edilerek, işlevsel büyüklüğün otomatik olarak ölçülmesi sağlanmıştır. Her iki çalışma kapsamında harcanan süreler kaydedilmiş, ayrıca karşılaştırma amacıyla geliştirme sürecinin farklı evrelerinde manuel ölçme gerçekleştirilmiş ve harcanan süreler kaydedilmiştir. Durum Çalışması C kapsamında ise geliştirme süreci tamamlanmış bir yazılım üzerine Yazılımcı 2 tarafından Measurement Kütüphanesi entegre edilerek otomatik işlevsel büyüklük ölçme işlemi gerçekleştirilmiş, bu sırada harcanan süre değeri kaydedilmiştir. Sonrasında aynı uygulama için manuel ölçme yapılmış ve bu

sırada geçen süre de yine kayıt altına alınmıştır. Sonrasında her iki ölçüm sonucu alınan değerler karşılaştırılmış ve değerlendirmeler yapılmıştır.

Durum Çalışmaları sonucunda elde edilen veriler, Measurement Kütüphanesi kullanılarak gerçekleştirilen otomatik ölçme işlemine ilişkin bazı çıkarımlar yapmamızı sağlamıştır. Bu çıkarımlara aşağıda yer verilmiştir:

1. Measurement Kütüphanesi'nin kullanımı için belirlenmiş kısıtlar çerçevesinde gerçekleştirilen otomatik ölçme işlemi sonuçları manuel yapılan işlem sonuçlarıyla kıyaslandığında, sonuçların birbirleriyle aynı olduğu görülmüş olup, kütüphane kullanılarak alınan ölçüm sonuçlarının güvenilir olduğu anlaşılmıştır.

2. Measurement Kütüphanesi'nin kullanılabilirliğinin, kütüphaneyi kullanan yazılımcının konu hakkındaki bilgisi ve tecrübesinden büyük oranda etkilenmediği, işlevsel büyüklük ölçümü ve Measurement Kütüphanesi konularında bilgisi ve tecrübesi bulunmayan bir yazılımcının kütüphaneyi yazılım geliştirme sürecinin en başından itibaren kullanması sonucunda, manuel ölçme işlemine göre 4,5 kat maliyet kazancı sağlandığı belirlenmiştir.

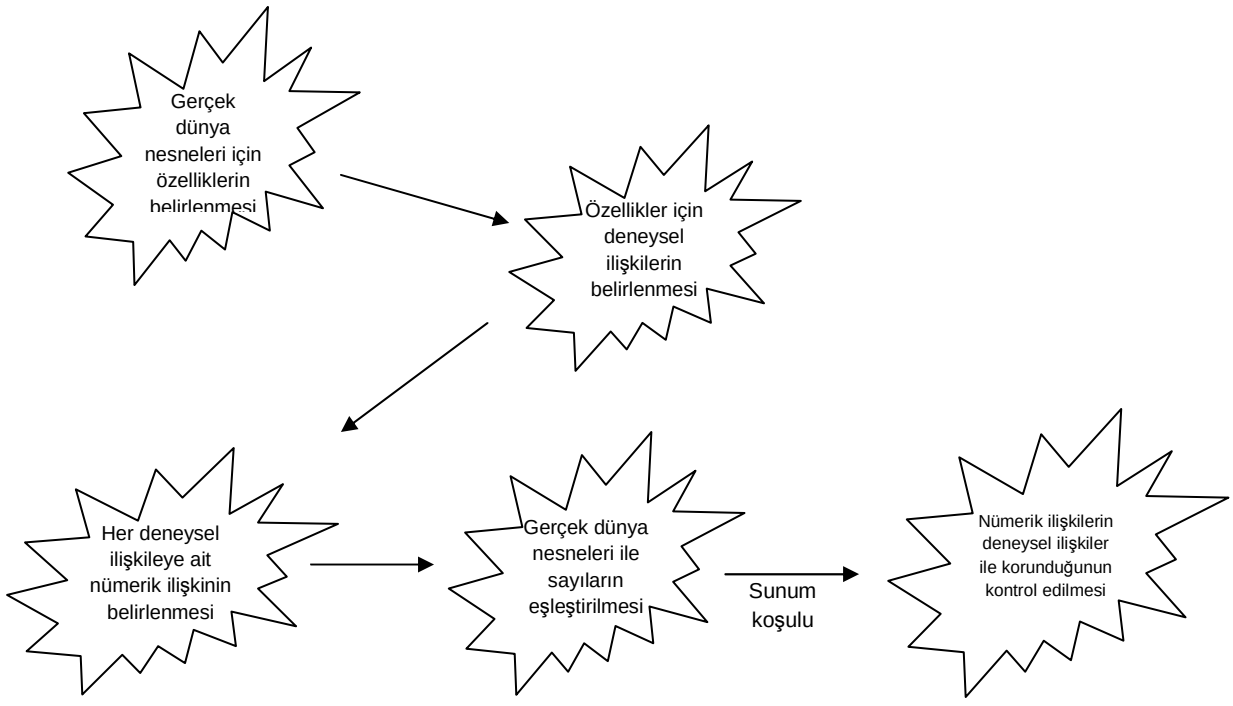
3. Measurement Kütüphanesi'nin geliştirme süreci sona ermiş bir uygulamaya, uygulama konusunda bilgisi olmayan bir yazılımcı tarafından entegre edilmesiyle gerçekleştirilen otomatik ölçme işleminin, aynı uygulama için manuel olarak yapılan ölçme işlemine göre daha maliyetli olduğu ortaya çıkarılmış, benzer durumlar için kullanılması durumunda maliyet-etkin olmayacağı ortaya konmuştur.

2. ÖN BİLGİ

2.1. Ölçme Kavramı

Ölçme kavramı, günlük hayat ile iç içe geçmiş bir kavramdır. Günlük hayatın her anında farkında olmadan ölçme işlemi uygularız. Satın aldığımız bir ürün için ödeme yaparken, evden işe giderken gideceğimiz yol ve harcayacağımız vakti hesaplarken, günlük programımızı yaparken ve yapacağımız işlerimizi planlarken ölçme işleminden yararlanırız. Ölçme işlemi, dünyamızı anlama, çevremizle iletişim kurma ve yaşamımızı iyileştirme konularında bizlere yardımcı olmaktadır [27].

Ölçme, gerçek dünyadaki varlıklara ait nitelikleri açıkça belirlenmiş kurallara göre tanımlama amacıyla bu varlıklara ait niteliklere sayıların ve sembollerin atanması sürecidir [27]. Ölçme sayesinde karşılaştırma ve değerlendirme yapmamız mümkün olmaktadır. Bir cismin uzunluğunu ve kısalığını, bir odanın büyüklüğünü ve küçüklüğünü, bir cismin uzaklığını ve yakınlığını ölçme sayesinde bu kavramlara yüklenen anlamlar ile değerlendirebiliyoruz.



Şekil 2.1 Standart Ölçme İşleminin Temel Aşamaları [27]

Bilimsel olarak yapılan çalışmaların tamamında bir ölçme ve değerlendirme adımı bulunmaktadır. Ölçme işlemi sayesinde karşılaştırmalar, değerlendirmeler ve

iyileştirmeler yapılabilir. Aksi halde yapılan çalışmalar öznel birer iddiadan ileri gidemezler ve bu sebeple bilimin gelişimine katkıda bulunmaları imkansız bir hal alır. Galilei “Ölçülemeyen şeyleri ölçülebilir yapın” sözüyle, tam da bu noktada çevremizdeki varlıkları ve gelişen olayları ancak ölçme ile anlamlandırabileceğimize vurgu yapmış, halihazırda ölçemediğimiz kavramları da ölçülebilir hale getirmemiz için çaba göstermemiz gerektiğini ifade etmiştir [27].

Birçok disiplinde yapılan çalışmalar ve geliştirilen projeler ölçme işlemi ile iç içe geçmiştir. Bir mimarın, fizikçinin ve ekonomistin yaptığı çalışmalar ilgili metriklerden bağımsız düşünülemez [27]. Ancak yazılım mühendisliğinde metriklerin kullanımı, diğer disiplinlerdeki kullanımına nazaran çok yenidir. Halen yazılım mühendisliği alanında gerek yazılımcı, gerek proje yöneticisi gerekse araştırmacı olarak çalışan birçok kişi, yazılım mühendisliğinde ölçme kavramına oldukça yabancıdır. Bu kavrama aşina olan kişilerden bazıları özellikle zaman ve işgücü kısıtları sebebiyle bu konunun üzerine düşmemekte ve ikinci plana atmakta, bazıları ise halen güvenilirlik, kalite, kullanılabilirlik ve sürdürebilirlik gibi kavramların kolay bir şekilde ölçülebilir olmadıklarını düşünmektedirler [27].

2.1.1. Yazılım Mühendisliği’nde Ölçme Kavramı

IEEE bünyesindeki Bilgisayar Grubu’nun tanımına göre Yazılım Mühendisliği[28],

(1) Yazılımın geliştirilmesi, işletilmesi ve idamesi için sistematik, disiplinli, ölçülebilir bir yaklaşımın uygulanmasıdır, yani, yazılıma mühendisliğin uygulanmasıdır.
(2) (1). Maddede belirtilen yaklaşımlara ilişkin çalışmalardır.

Tanımdan da anlaşılacağı üzere yazılım mühendisliği, sistematik, disiplinli ve ölçülebilir bir yaklaşımdır. Ancak yazılım projelerinden ve yapılan çalışmalardan anlaşıldığı üzere yazılım mühendisliğinin ‘*ölçülebilir*’lik özelliği göz ardı edilmektedir. Literatürde, bütçelerini ve zamanlarını aşan birçok yazılım projesi bildirilmektedir [27]. Bunun sebebi başarısız proje yönetimi süreçleridir. Bu sorun ancak yazılım mühendisliği kavramlarının belirlenecek standartlar kapsamında ölçülebilir hale getirilmesi ve ölçme işleminin getirdiği avantajlardan proje yönetiminin her aşamasında faydalanılması ile aşılabilecektir.

Alain Abran’a göre yazılım ölçmenin yazılım endüstrisinin bir parçası haline gelebilmesi için ölçme, yazılım endüstrisinde teknik ortam ve yönetim bağlamı ile

entegre hale gelmeli ve yazılım mühendisliğinde ölçmenin birçok kapsamda başarılı sonuçlar verdiği kanıtlanmalıdır [29].

Bilimsel çalışmalarda önerilen metot ve yaklaşımlar bir modelin geçerliliğini ya da bir teorinin doğruluğunu iddia eder. Teorinin pratiğe dökülmesi için ise ölçme devreye girer. Diğer birçok disiplini ölçmeden ayrı düşünemezken, günümüzde halen yazılım mühendisliği için ölçme lüks olarak algılanmakta, yapılan ölçümler ise seyrek, tutarsız ve tamlıktan uzak olmaktadır. Bu sebeple yapılan bu ölçümlere güvenerek onları kendi projelerimize uygulamamız mümkün olmayacaktır, ayrıca kendi ortamımızda aynı ölçme işlemini tekrarlayabileceğimiz nesnel bir çalışma yapmamız imkansızdır. Yazılım mühendisliğinde ölçme eksikliği titiz ve dikkatli bir yaklaşımın sergilenmemesinden kaynaklanmaktadır [27].

Yazılım mühendisliğinin, diğer mühendislik disiplinlerine ve bilimsel disiplinlere göre alışılmadık özelliklerinden biri, karar almada nicel verilerin genel kullanım eksikliğidir. Bunun belirtileri, uygulayıcılar için kullanabilecekleri çok sınırlı sayıda kabul edilmiş ve uluslar arası standart olarak tanınabilecek olgunlukta yazılım ölçme birimi olması ve bu konuda çok az sayıda deneysel çalışmanın bulunmasıdır [29]. Ortak bir standart oluşturulamaması, bu konuda farklı kişi ve gruplarca yapılan çalışmaların kıyaslanabilirlikten uzak olmasına sebep olmaktadır. Kıyaslanamayan deneysel sonuçların ise daha iyiye erişme noktasında yazılım mühendisliğinde ölçme disiplinine gerekli katkıyı sağlamalarının zor olduğu değerlendirilmektedir. Standartlaşma konusunda farklı bakış açılarına sahip araştırmacıların ortak bir amaca ulaşmak amacıyla koordineli çalışmaları büyük önem taşımaktadır.

2.1.2. Yazılım Mühendisliği'nde Ölçme Amaçları

Yazılımda ölçme projenin cari durumundan haberdar olabilmek, projenin planlandığı gibi gidip gitmediğini değerlendirebilmek, eğer bir sorun var ise nereden kaynaklandığını zamanında fark ederek gerekli aksiyonları alabilmek, daha önce yapılan ölçme kayıtlarından yararlanarak gidişe hata dayalı öngörüde bulunabilmek ve sonraki çalışmalar için referans alınabilecek kayıtlar oluşturabilmek gibi birçok amaca hizmet etmek üzere gerçekleştirilebilir.

Her ölçme eyleminin açıkça tanımlanmış ve kolayca anlaşılabilen belirli bir hedefi olmalı ve belirli bir ihtiyacı karşılamaya yönelik gerçekleştirilmesi gerekmektedir. Ölçme hedefleri yöneticiler, geliştiriciler ve kullanıcıların bilmeleri gerekene bağlı

olarak özelleşmiş olmalıdır. Ölçme bilgileri toplandığında nasıl kullanılacaklarına ilişkin bilgiyi bize bu hedefler verecektir [27].

Fenton ve Pfleeger yazılım geliştirme sürecinde ölçmenin üç temel aktivite için oldukça önemli olduğunu belirtmişlerdir. Bunlar anlama, kontrol etme ve geliştirmedir [27].

İlk olarak, geliştirme ve bakım sürecinde neler olduğunu anlamamıza yardımcı olan ölçümler vardır. Bu ölçümler yardımıyla projenin cari durumunu değerlendirebilir, ayrıca cari durumu temel alarak gelecek davranışlar için hedefler belirleyebiliriz. Bu kapsamda yapılan ölçümler, etkiledikleri aktiviteler ve nitelikler arasındaki ilişkiyi daha iyi anlamamızı, böylece süreç ve ürüne ilişkin görüşlerimizin daha net olmasını sağlar. İkinci olarak, ölçme bizlere projelerimizde neler olduğunu kontrol etme imkanı verir. Bu ölçümler sayesinde hedeflerimizi ve ilişkiler konusundaki anlayışımızı kullanarak, neyin olmak üzere olduğunu tahmin edebilir ve ürün ve süreçler üzerinde hedeflerimize ulaşmamıza yardımcı olan değişiklikleri gerçekleştirebiliriz. Üçüncü ve son olarak, ölçme süreç ve ürünlerimizi geliştirmemiz yönünde bizleri destekler [27]. Örneğin proje geliştirme sürecinin herhangi bir anında elde edilen ürünün nitelikleri ölçülüp, geliştirme öncesi yapılan öngörüler veya aynı nitelikte daha önce yapılmış çalışmalarda elde edilen ölçüm verileri ile karşılaştırılarak değerlendirmeler yapılabilir, ürünlerin geliştirilmesi sağlanabilir.

2.2. Yazılım Boyutlarının Ölçülmesi

Yazılım boyutunun birçok yönü vardır. En önemli yönleri ‘uzunluk’, ‘işlevsellik’, ‘karmaşıklık’ ve ‘tekrar kullanılabilirlik’tir [27]. Uzunluk ortaya çıkan ürünün fiziksel büyüklüğünü belirtir. İşlevsellik ise ürünün kullanıcıya sağladığı işlevlerin büyüklüğüdür. Karmaşıklık bakış açısına göre farklı şekillerde yorumlanabilir (problem karmaşıklığı, algoritma karmaşıklığı, yapısal karmaşıklık, idrak karmaşıklığı vb). Yeniden kullanılabilirlik ise, ortaya çıkan ürünün ne kadarının yeni geliştirildiği, ne kadarının önceki çalışmalardan yararlanılarak tamamlandığı ile ilgili bir büyüklüktür [27].

2.2.1. Uzunluk

Yazılım geliştirme süreci iş ürünlerinden ‘Belirtim’, ‘Tasarım’ ve ‘Kod’ için uzunluk değerlerinin bilinmesi yapılacak değerlendirmeler için yararlı olacaktır. Belirtimlerin uzunluğunun ölçülmesi tasarımın olası uzunluğu konusunda kullanışlı bir göstergedir,

aynı zamanda kod uzunluğunun tahmini için de kullanılır. Benzer şekilde, erken ürünlerin uzunluğu, sonrakiler için gerekli olacak iş gücü miktarının belirlenmesi için kullanılabilir [27].

2.2.1.1. Kod

Geleneksel kod uzunluk ölçümlerinden en yaygın kullanılan yöntem Kod Satır Sayısı (KSS) yöntemidir [27]. Kod satır sayısı, bir satırdaki komut sayısı ve komut bölünmelerinden bağımsız, yorum satırı ve boş satırlar hariç program uzunluğudur. Bu uzunluk, program başlıkları, tanımlamalar, çalıştırılabilir ve çalıştırılabilir olmayan belirtimleri içermektedir [30].

Kod satır sayısı ölçümü, sayımın kapsamı ve tanımı açısından belirsizlikler içermektedir. Bir fiziksel kod satırının bir komuta eş değer olduğu Assembler programlamanın ilk zamanlarında, kod satır sayısı tanımı oldukça açıktı. Yüksek seviyeli dillerin kullanılmaya başlaması ile bu birebir ilişki geçersiz kalmıştır. Kod satır sayısı ile komut sayılarının arasındaki farklar ve bunların diller arasında da değişiklik göstermesi, kod satır sayısı ölçümünde çeşitlenmeyi beraberinde getirmiştir [31].

Bunun dışında farklı programcıların farklı kodlama alışkanlıkları kod satır sayısı ölçümünde ortak bir standardın sağlanmasını zorlaştırmaktadır. Kimi programcıların okunabilirliği arttırmak amacıyla boş satırlar bırakması ya da diğer bir programcının aynı satırda birden fazla komuta yer vermesi, bu belirsizliğin kaynağına ilişkin örnek olarak verilebilir.

Galorath ve Evans'ın çalışmalarında [32] ele aldıkları "kod satır sayısı" metriğinin avantaj ve dezavantajlarına Çizelge 2.1'de yer verilmiştir.

Çizelge 2.1 Kod Satır Sayısı (KSS) kullanımının avantaj ve dezavantajları [32]

Avantaj	Dezavantaj
Kolay hesaplanabilir	Kodun nasıl yazıldığına doğrudan bağlıdır.
Sezgisel	KSS'yi oluşturan etmenlerin birden fazla tanımı bulunmaktadır.
Tanecikli	Kullanılan teknoloji uyarınca farklılık göstermektedir.
Sürecin doğal bir sonucu olarak ortaya çıkar.	Yazılım geliştirme sürecinin sonunda kullanılabilir hale gelir.
	Bazı dillerin otomatik sayımı güç olmaktadır.

Bu alanda yapılan çalışmalar kod satır sayısının belirlenmesi için farklı yaklaşımlar ortaya koymuştur. Boehm'in Software Engineering Economics adlı kitabında yer alan kod satır sayısı sayma metodu, çalıştırılabilir satırlar, veri tanımlamaları ve yorum satırlarını da içermektedir [33]. Grady ve Caswell'in 1987 yılında yaptıkları çalışmada Hawlett-Packard'ın kod satır sayısı tanımını, programdaki yorum satırları ve boş satırlar dışındaki tüm ifadeler olarak yaptığını bildirmişlerdir [34]. Hawlett-Packard'ın yaptığı bu tanım geniş biçimde kabul görmüştür. Bu tanımdaki yorumsuz kod satır sayısı NCLOC ("Non Commented Lines of Code"), kimi zaman etkin satır sayısı ELOC ("Effective Lines of Code") olarak anılmıştır.

Fenton ve Pfleeger kod satır sayısı ölçümündeki belirsizliğin ortadan kalkması için şu kavramların nasıl ele alındığının açıklanması gerektiğini belirtmişlerdir [27]:

- Boş satırlar,
- Yorum satırları,
- Veri tanımlamaları,
- Birden fazla komut içeren satırlar.

Şirketler kod satır sayısı verisini kullanarak farklı değerlendirmeler yapabilirler. Bazıları yaptıkları projeler arasında büyüklük, etkinlik ve maliyet karşılaştırması yapabilmek için, bazıları aynı proje için farklı modüller arasında karşılaştırmalar yaparak anlamlı sonuçlar çıkarabilmek için kod satır sayısı metriğini kullanabilirler.

Uzunluk ölçümü için farklı alternatifler de ortaya atılmıştır [27]. Bu yaklaşımlara aşağıda yer verilmiştir.

a) Yazılım büyüklüğü, yazılımın bilgisayar belleğinde kapladığı alanın bayt cinsinden değeri olarak ölçülebilir. Çok daha kolay elde edilebilecek bir büyüklüktür.
b) Yazılım uzunlukları karakter bazında da ölçülebilir. Birçok araç bunu varsayılan olarak gerçekleştirmektedir.

Nesneye yönelik programlama yaklaşımının ortaya çıkmasıyla birlikte, farklı yazılım uzunlukları hesaplama yaklaşımları önerilmiştir. Pflegger nesne ve metotların sayısının çok daha doğru üretkenlik tahminlerini beraberinde getireceğini belirtmiştir [35]. Lorenz, IBM'deki çalışmasında ortalama bir sınıfın 20 metot içerdiğini, ortalama bir metodun ise Smalltalk için 8, C++ için ise 24 satır kod içerdiğini belirlemiştir. Ayrıca uzunluğun sistem ve uygulama türüne göre de değişkenlik gösterdiğini belirtmiştir [36].

2.2.1.2. Belirtimler ve Tasarım

Yazılım yaşam döngüsünün erken evrelerinde belirtim ve tasarım dokümanları genellikle yazı, grafik ve özel matematiksel diyagramlar ile semboller içerirler. Kod boyutunun ölçümü için ilgili belirtim-tasarım dokümanı içerisinde sayılacak atomik nesneler belirlenir. Ancak belirtim ve tasarımlar, uzunlukla kıyas yapılamayan yazı ve diyagramlar içerebilmektedir. Bu tür durumlarda karşılaştırılabilirliği sağlamak için, atomik öğe olarak yazı ve diyagramların sayfalardan oluştuğu varsayımıyla sayfalar belirlenebilir. Sayfa sayısı birçok farklı doküman çeşidi için kullanılabilirdir ve sektörde en çok kullanılan yöntemdir. Ancak aynı doküman içinde hem yazı hem diyagram olabileceği, bunun yanında diyagramların çeşidinin de kendi içinde farklılaşabilmesi sebebiyle atomik öğe olarak daha genel seçimler yapılması gerekliliği ortaya çıkmıştır [27].

2.2.2. Tekrar Kullanılabilirlik

Program geliştiriciler, dördüncü nesil diller ve diğer işgücü-tasarruf teknikleri bazı görevlerin tekrarlanmasından yararlanmaktadırlar. Başka bir ürün veya proje için geliştirilmiş bir alt sistemin, sonra geliştirilen ürün ve projeler için tekrar geliştirilmesi ihtiyacı duyulmamaktadır. Bu sayede sıklıkla kullandığımız işletim sistemleri, derleyiciler, VTYS'ler tekrar tekrar yazılmadan kullanılabilmektedir. Yazılımların yeniden kullanılabilirliği, verimlilik ve kaliteyi arttırır ve yeni problemler üzerine yoğunlaşabilmemizi sağlar [27].

Yazılımların tekrar kullanılabilirliği fikri ilk olarak 1968 yılında NATO Yazılım Mühendisliği toplantısında ortaya çıkmıştır. O dönemde içinde bulunulan yazılım krizi için önerilen çözümlerden biri de yazılımların tekrar kullanılmasıdır [37]. Konferansta ortak bir kütüphane sayesinde önceden geliştirilen yazılımlar kullanılarak yazılım geliştirme maliyetlerinin düşürülebileceği fikri Mcllroy tarafından ortaya atılmıştır [38].

Yazılımın yeniden kullanılması yalnız kaynak kod parçalarında değil, yazılım yaşam döngüsünün her ürünü üzerinde uygulanabilir. Dolayısı ile geliştiriciler, gereksinim dokümanlarının, sistem belirtimlerinin, tasarım yapılarının ve diğer geliştirim dokümanlarının yeniden kullanılabilirliğinden yararlanabilirler [39]. Jones yazılım projelerinde potansiyel yeniden kullanılabilir olan on farklı nokta belirlemiştir [40]. Bu noktalar Çizelge 2.2'de gösterilmiştir.

Çizelge 2.2 Yazılım Projelerinin Yeniden Kullanılabilecek Unsurları [40]

1.Mimari	6.Taslaklar
2.Kaynak Kod	7.Kullanıcı arayüzleri
3.Veri	8.Planlar
4.Tasarım	9.Gereksinimler
5.Dokümantasyon	10.Test Senaryoları

Yazılım ölçme faaliyetlerimiz içine tekrar kullanılan ürünleri de eklememiz gerekmektedir. Tekrar kullanılan kodların boyutları tekrar kullanılan kod ile neyi anlatmak istediğimiz ile ilgilidir. Kimi zaman programın bütününü kullanırken, kimi zaman ise programın bir kısmı tekrar kullanılmaktadır. Kimi zaman ise programın bir kısmı güncellenerek kullanım gerçekleştirilir [27].

NASA/Goddard's Software Engineering Laboratory tarafından yazılımın hangi oranda yeniden kullanıldığı ile ilgili "yeniden kullanılırlık derecesi" ("extent of reuse") belirlenmiştir [27].

1. Kelimesi kelimesine yeniden kullanım ("reused verbatim"): Birimdeki kodun hiç değiştirilmeden kullanılmasıdır.
2. Az değiştirilmiş ("slightly modified"): Birimdeki kod satır sayısının %25 ve daha azının değiştirilmiş olmasıdır.
3. Geniş anlamda değiştirilmiş ("extensively modified"): Programın, kod satır sayısının %25 ve daha fazla oranda değiştirilmesidir.

4. Yeni (“new”): Kodun hiçbir kısmı daha önce geliştirilmiş birimden gelmemektedir.

2.2.3. İşlevsellik

Birçok yazılım mühendisine göre, yazılım uzunluğu yanıltıcı olmakta ve ürünün doğasında olan işlevsellik, ürün boyutu için daha iyi bir resim çizmektedir. Özellikle geliştirme sürecinde erken ortaya çıkan iş ürünleri üzerinden işgücü ve süre tahmini yapabilen yöntemler sıklıkla işlevsellik ölçümü için tercih edilmektedirler. Farklı bir nitelik olarak işlevsellik sezgisel olarak, tamamlanan üründeki işlev miktarını veya tanımlamadaki ürünün nasıl olması gerektiğini yansıtır [27]. Yazılım uzunluğu daha çok geliştirici perspektifinden bakış açısını yansıtırken, işlevsellik kullanıcı bakış açısını yansıtmaktadır.

Yazılım ürünlerinin işlevselliğinin ölçülmesi ile ilgili yapılan çalışmalardan en önemli üç tanesi Albrecht’in işlev puanları [1], DeMarco’nun belirtim ağırlıkları [41] ve COCOMO 2.0 [42] nesne puanları yaklaşımlarıdır. Bu yaklaşımların hepsi belirtim dokümanlarının işlevselliğini ölçer, ancak her biri yaşam döngüsünün sonraki evrelerinde ortaya çıkan ürünlere de uygulanarak boyut tahminlerinin, bu nedenle de maliyet ve verimlilik tahminlerinin iyileştirilmesini sağlar [27].

2.2.3.1. Albrecht Yaklaşımı

Yazılım mühendislerinin birçoğu için yazılım büyüklüğü “kod satır sayısı” olarak ifade edildiği zamanlarda, kod satır sayısı metriği kullanılan programlama diline göre çok farklı anlamlara sahipti. Albrecht bu standarttan uzak yaklaşımın yerine projede kullanılan programlama dilinden bağımsız olan bir yaklaşım önermiştir.

“İşlev puan” olarak adlandırılan bu metrik, geliştirilen uygulamanın sağladığı işlevselliğin büyüklüğünü hesaplamayı amaçlar. İşlev puanlar hesaplanırken öncelikle aşağıda verilen bileşenler üzerinden “Düzeltilmemiş İşlev Puanlar” (“Unadjusted Funtion Points-UFP”) hesaplanır. Longstreet yaptığı çalışmada, “Düzeltilmemiş İşlev Puan” hesabı için temel alınan bileşenleri şu şekilde açıklamıştır [43]:

Harici Girdiler (“External Inputs –EI”): Verinin dışarıdan içeriye doğru hareket ettiği temel süreç olarak tanımlanmaktadır. Söz konusu verinin bir girdi ekranından veya başka bir uygulamadan gelmesi mümkün olmakla birlikte veri, bir veya birden fazla

dahili mantıksal dosyalar üzerinde değişiklik yapılması amacıyla kullanılabilir. Veri kontrol bilgisi veya iş bilgisi olabilmektedir. Verinin kontrol bilgisi olması durumunda, dahili mantıksal dosyalarda herhangi bir güncelleme gerçekleştirilmemektedir.

Harici Çıktılar ("External Outputs–EO"): Verinin içeriden dışarıya doğru geçiş hareketinden meydana gelen temel süreç olarak tanımlanmaktadır. Harici çıktılar, dahili mantıksal dosyalar üzerinde güncellemeler gerçekleştirebilmektedir. Verinin oluşturduğu raporlar veya çıktılar diğer uygulamalara aktarılabilmektedir. Söz konusu rapor ve çıktılar, bir veya birden fazla dahili mantıksal dosyadan üretilmektedir.

Harici Sorgular ("External Inquiry–EQ"): Verinin bir veya birden fazla dahili mantıksal dosya/harici arayüz dosyalarından alınması sonucu ortaya çıkan girdi ve çıktı bileşenleri olarak tanımlanmaktadır. Veri girişi herhangi bir dahili mantıksal dosyada güncellemeye neden olmamakta ve veri çıkışı esnasında bu tarz bir veri oluşmamaktadır.

Dahili Mantıksal Dosyalar ("Internal Logical Files–ILFs"): Uygulama sınırları dahilinde yer alan, tespit edilebilir ve mantıksal olarak ilişkili bir grup veri olarak tanımlanmaktadır. Harici girdiler tarafından değiştirilebilmektedir.

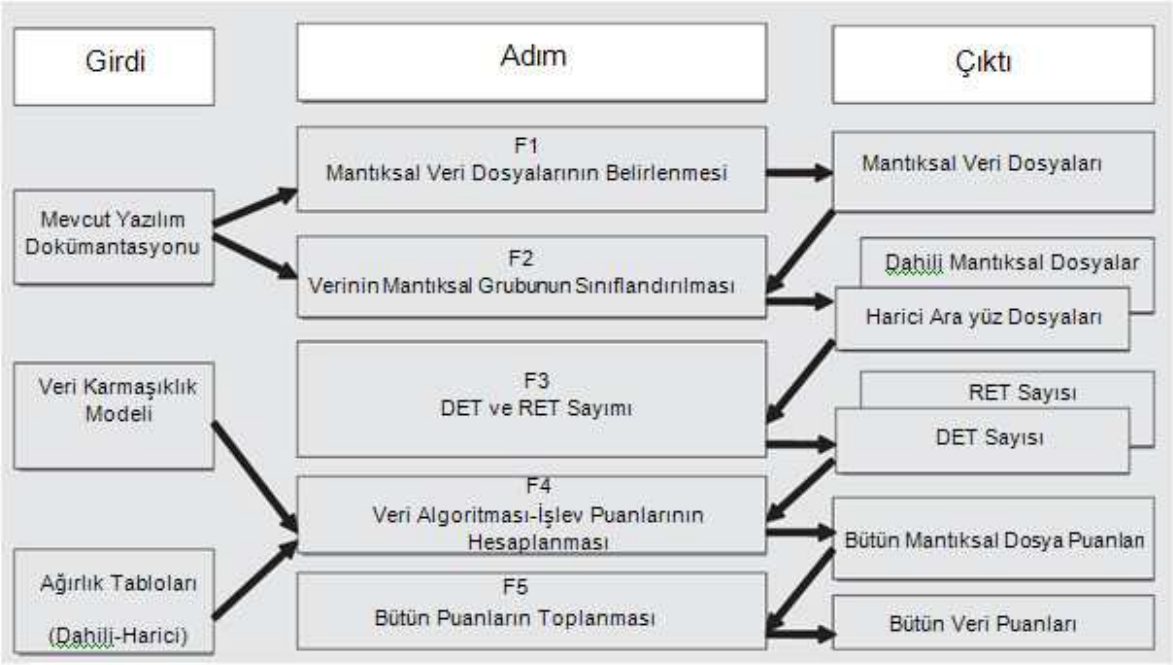
Harici Arayüz Dosyaları ("External Interface Files–EIFs"): Sadece referans olarak kullanılan ve mantıksal olarak ilişkili bir grup veri olarak tanımlanmaktadır. Veri uygulamanın tamamen dışında yer almakta ve başka bir uygulama tarafından değiştirilmektedir. Herhangi bir harici arayüz dosyası başka uygulamalar için dahili mantıksal dosya olmaktadır.

Bu bileşenlerden elde edilen değerler, karmaşıklıklarına göre, düşük, orta ve yüksek olmak üzere üç farklı seviyeye ayrılırlar ve daha sonra ilgili seviye için belirlenmiş karmaşıklık faktörü ile ağırlıklandırılırlar. Bu işlem tüm bileşenlere ait değerlere uygulandıktan sonra elde edilen değerler toplanarak "Düzeltilmiş İşlev Puan" hesaplanmış olur.

Alain Abran, işlev puan hesaplama adımlarını kitabında Çizelge 2.3 teki gibi tanımlamış ve bu işlemleri Şekil 2.2 deki gibi görselleştirmiştir [29].

Çizelge 2.3 İşlev Puan Hesaplama Adımları [29]

Sıra	*Adım Açıklaması*
F1	Bu adımda yazılım fonksiyonlarına ait mevcut bilgiler (yazılım dokümanlarından veya yazılımın kendisinden elde edilen) girdi olarak kullanılmaktadır. Bu bilgi aracılığı ile ölçümü gerçekleştiren kişi, mantıksal veri dosyalarını belirlemektedir.
F2	Ölçümü gerçekleştirilen uygulama ile diğer harici uygulamalar arasındaki sınırların belirlenmesi için birinci adımdaki bilgi kullanılmaktadır. Bu adım sonucunda mantıksal veri dosyaları, harici mantıksal veri dosyası veya harici arayüz dosyası olarak sınıflandırılmaktadır.
F3	İkinci adımda tamamlanan sınıflandırma bilgisi bu adımda; her bir dosya için Veri Elemenı Türü sayısının (“Data Element Type-DET”), dahili mantıksal dosyalar için Kayıt Elemenı Türü sayısının (“Record Element Type-RET”) ve harici arayüz dosyaları için Referanslanan Dosya Türünün (“File Types Referenced-FTR”) sayılması ve belirlenmesi amacıyla kullanılmaktadır.
F4	Ölçümü gerçekleştiren kişi aşağıda yer alan beş girdi aracılığı ile İşlev Puan veri algoritmasını uygulamaktadır: • Dahili mantıksal dosyaların ve harici arayüz dosyalarının listesi • Kayıt elemanı türü listesi (RET) • Veri elemanı türü listesi (DET) • Verinin karmaşıklık modeli (örneğin; karmaşıklık modelinin iki boyutlu ekseni ile birlikte yapısı) • Dosyaların tabloda yer alan adım-fonksiyonu (“step-function”) ile uyumlu ağırlıkları (harici ve dahili)
F5	Dosyaların her birinden (hem harici hem dahili dosyalar) elde edilen puanlar toplanarak, veriye ait Düzeltilmemiş İşlev Puan oluşturulmaktadır.

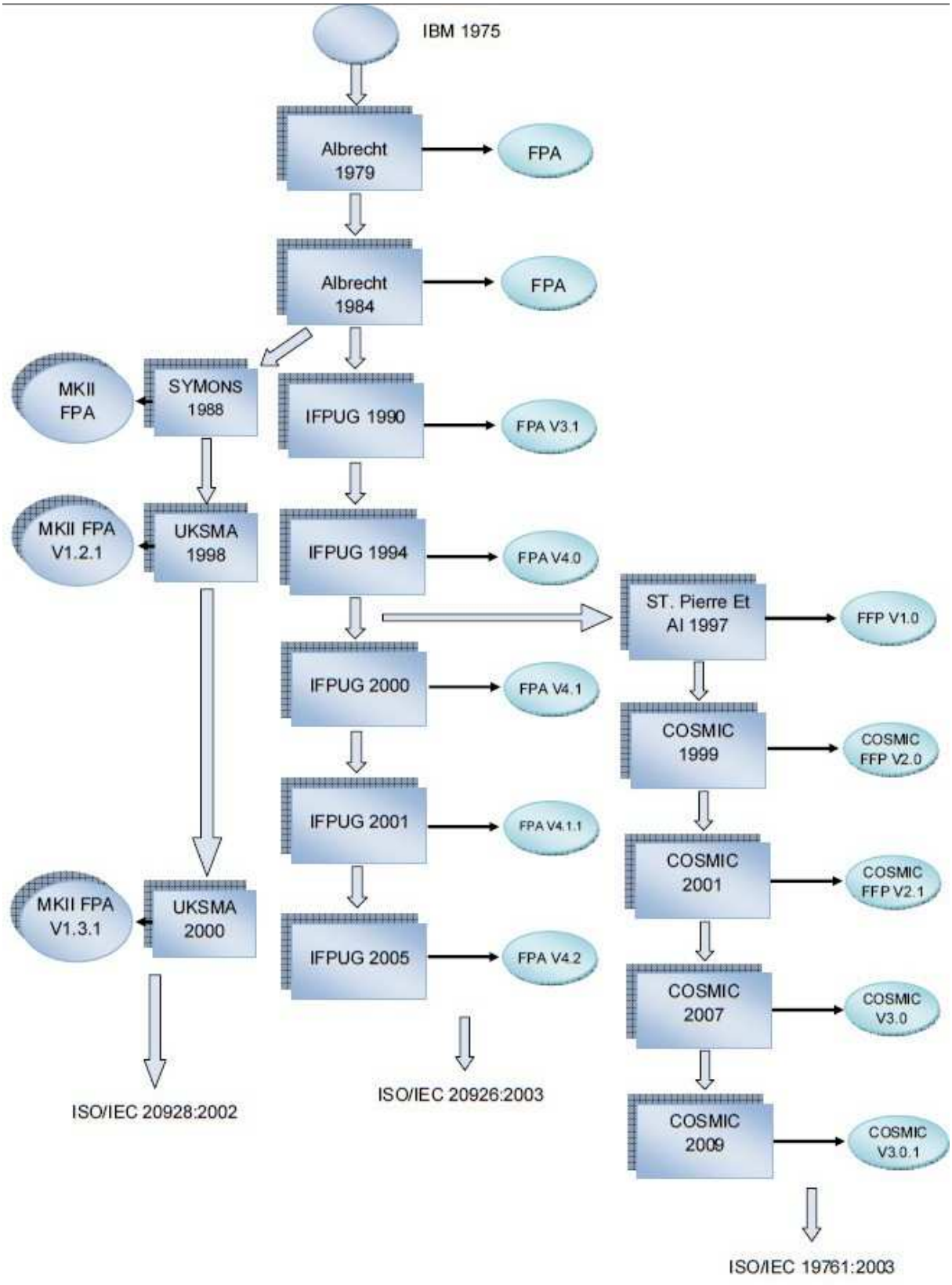


Şekil 2.2 İşlev Puan Hesaplama Adımları [29]

Albrecht bu yöntemi teknoloji bağımsız bir büyüklük ölçümü olarak önermiştir. Ancak işlev puan ölçümünde karşılaşılan birtakım problemler vardır. Bu yöntemi kullanan araştırmacıların bunları göz önünde tutmaları gerekmektedir [27]. Bu problemlerden birkaçı aşağıda verilmiştir:

1. Teknik karmaşıklık faktörü hesaplamasındaki öznellik problemleri
2. Ağırlıklandırma konusundaki öznellikler
3. Yaşam döngüsünün erken evrelerinde kullanılma ile ilgili problemler (erken evrelerde kullanılabilmesi için tüm yazılım sisteminin belirtimini gerektirir, kullanıcı gereksinimleri dokümanı yeterli gelmez.)
4. Değişen gereksinimler sonucu yaşanan problemler (yazılım geliştirme yaşayan bir süreçtir ve geliştirim sırasında yeni gereksinimler ortaya çıkabileceği gibi aynı gereksinim için öngörülemeyen hareketler gerçekleşebilir.)

Albrecht tarafından tanımlanan işlev puan modeli, zaman içinde, yapılan çalışmalarla yenilenmiş ve geliştirilmiştir. 1986'da IFPUG ("International Function Point Users Group") kurulmuştur. IFPUG işlev puan sayımı için kılavuz hazırlamış ve ilerleyen yıllarda meydana gelen gelişmelerle birlikte bu kılavuzu güncellemiştir. Son olarak kılavuzun 4.3 sürümü yayımlanmıştır [8]. İşlev puan sayımı için geliştirilen yaklaşımlar tarihsel sırası ile Levesque et al. tarafından Şekil 2.3'teki gibi gösterilmiştir [44].



Şekil 2.3 İşlev Puan Yaklaşımının Tarihsel Gelişimi [44]

2.2.3.2. COCOMO 2.0 Yaklaşımı

COCOMO temelde işgücü ve maliyet tahmini yapabilme amacıyla ortaya çıkmış bir yaklaşımdır. 1970'li yıllarda Barry Boehm'in ortaya attığı bu yaklaşım, o güne kadar yazılım mühendisliğine ekonomi bilimi açısından bakan ilk yaklaşım olması açısından büyük önem taşımaktadır [3]. COCOMO yaklaşımı aşağıda verilen denklem temelinde çalışmaktadır;

$$E = aS^bF$$

Denklemde E, adam ay cinsinden işgücünü, S kod satır sayısı cinsinden büyüklüğü, F ise düzeltme faktörünü göstermektedir. a ve b değerleri ise geliştirilen yazılımın türüne göre belirlenen geliştirme moduna göre farklı değerler almaktadır. Organik sistem, bankacılık sistemleri gibi veri işleme özelliği olan sistemlerdir. Gömülü sistem, gerçek zamanlı çalışan yazılım içeren, donanım tabanlı sistemlerdir. Yarı müstakil sistemler ise bu iki sistem arasında kalan sistemlerdir.

COCOMO 2.0 yaklaşımında, yöntemin uygulama geliştirme sürecinin daha erken evrelerinde kullanılabilmesi amacıyla, büyüklük değeri olarak standart COCOMO'da kullanılan kod satır sayısı yerine "nesne puan"lar kullanılmıştır [45]. Nesne puanları, uygulama içerisindeki ekranlar, raporlar ve üçüncü nesil dil bileşenlerinin sayılması ile hesaplanır. Sonrasında her nesne Albrecht'in işlev puan yaklaşımına benzer bir şekilde karmaşıklık seviyelerine göre kolay, orta ve zor olmak üzere sınıflandırılmaktadır. Boehm'in bu sınıflandırma için önerdiği yol, Çizelge 2.4'te verilmiştir. Sonrasında bileşenler ilgili karmaşıklık seviyesinin bir örneği için gerçekleştirme maliyeti temel alınarak ağırlıklandırılır. Bu ağırlıklandırma aynı çizelge üzerinde gösterilmiştir. Sonrasında ağırlıklandırılan bu nitelikler toplanarak toplam nesne puan değeri hesaplanır. Son olarak COCOMO 2.0 yönteminde, COCOMO'dan farklı olarak yeniden kullanım gözönüne alınarak nesne puanların bir kısmının önceki projelerden kullanıldığı varsayılarak nihai nesne puan aşağıda verilen formüldeki gibi hesaplanır [45]. Denklemde r önceki projelerden yeniden kullanılan nesne puan yüzdesini göstermektedir.

Yeni nesne puanları = (nesne puanları) x (100-r)/100

Çizelge 2.4 Nesne Puan Çıkarım Prosedürü[45]

Nesne Puan Hesaplama Adımları:

1. Adım - Nesne Sayılarının Belirlenmesi: Ekran sayısının, rapor sayısının ve uygulamanın içerdiği 3GL bileşenleri belirlenmektedir. Söz konusu nesnelerin standart tanımlarının ICASE ortamında olduğu varsayılmaktadır.

2. Adım - Tüm nesne örnekleri, kendilerine özgü boyut değerleri doğrultusunda "basit, orta, zor" olmak üzere sınıflandırılmaktadır. Sınıflandırma esnasında aşağıda yer alan tablo baz alınmaktadır.

Ekran				Rapor			
Görüntü-lenme Sayısı	Veri sınıfının kaynağı ve sayısı			Oturum Sayısı	Veri sınıfının kaynağı ve sayısı		
	Toplam <4 (<2 sunucu <3 istemci)	Toplam <8 (2/3 sunucu 3-5 istemci)	Toplam 8+ (>3 sunucu >5 istemci)		Toplam <4 (<2 sunucu <3 istemci)	Toplam <8 (2/3 sunucu 3-5 istemci)	Toplam 8+ (>3 sunucu >5 istemci)
<3	Basit	Basit	Orta	0 veya 1	Basit	Basit	Orta
3-7	Basit	Orta	Zor	2 veya 3	Basit	Orta	Zor
>8	Orta	Zor	Zor	4+	Orta	Zor	Zor

3. Adım - Aşağıdaki tablo uyarınca her hücrede yer alan numaraya bir ağırlık verilmektedir. Söz konusu ağırlıklar, nesnelerin karmaşıklık seviyesinin uyarlanması için gereken iş gücünü etkilemektedir.

Nesne Türü	Ağırlık-Karmaşıklık		
	Basit	Orta	Zor
Ekran	1	2	3
Rapor	2	5	8
3GL Bileşeni			10

4. Adım - Nesne Puanlarının Belirlenmesi: Tüm nesne örneklerine ait ağırlıklar toplanarak, NesnePuan'ı elde edilir.

5. Adım - Projede Hedeflenen Yeniden Kullanılabilirlik Yüzdesinin Belirlenmesi: Geliştirilecek yeni Nesne-Puanlar aşağıdaki formüle uygun olarak hesaplanmaktadır.

$$NOP = (Nesne\ Puan)(100 - \%Yeniden\ Kullanılabilirlik)/100$$

6. Adım - Üretkenlik oranı aşağıda yer alan formül ve tablo doğrultusunda hesaplanmaktadır.

$$PROD = NOP/(adam - ay)$$

Yazılımcının Tecrübesi ve Kapasitesi	Çok Düşük	Düşük	Nominal	Yüksek	Çok Yüksek
ICASE olgunluğu ve kapasitesi	Çok Düşük	Düşük	Nominal	Yüksek	Çok Yüksek
PROD	4	7	13	25	50

7. Adım - Tahmini adam-ay değeri (“person –month”) aşağıda yer alan formüle göre hesaplanmaktadır.

$$PM = NOP/PROD$$

2.2.3.3. DeMarco Yaklaşımı

DeMarco, gereksinim belirleme evresinde, “Yapısal Analiz” tanımının bileşenlerinden ürünün büyüklük ölçümünün yapılabileceği görüşünü ortaya atmıştır [41]. DeMarco yazılımları işlev-yoğun, veri-yoğun ve hibrit sistemler olarak gruplamıştır. Bu gruplamayı, veri modeli içerisindeki ilişki sayısını, veri akış diyagramı içerisindeki işlevsel nitelikteki basit yapıların sayısı ile oranlayarak yapmıştır. Bu oranın 0,7’nin altında olduğu sistemleri işlev-yoğun, 1,5’un üzerinde olanları veri-yoğun sistemler olarak adlandırmıştır.

İşlev yoğun sistemlerin ölçümünü “Function Bang” adını verdiği metrik ile gerçekleştirmiştir [41]. Bu metrik veri akış diyagramı üzerindeki işlevsel nitelikteki basit yapıların sayısını temel almıştır ve bu yapıların türüne göre ağırlıklandırma yapmıştır. Veri yoğun sistemlerin ölçümünü ise “Data Bang” adını verdi metrik ile ölçmüştür. Bu metrik ise varlık-ilişki modeli üzerindeki varlık sayısını temel almıştır. Temel varlık sayısı, her bir varlığın içerdiği ilişki sayısına göre ağırlıklandırılmaktadır [27].

2.2.4. Karmaşıklık

IEEE’ye göre karmaşıklık; uygulama ve bileşenlerin sahip olduğu, anlaşılması ve doğrulanması güç tasarım ve uyarlama seviyeleridir [28]. Evans ve Marciniak karmaşıklığı, sistem veya sistem bileşeninin aşağıda belirtilen faktörler ile belirlenen, güçlük seviyesi olarak tanımlamıştır [46].

- Arayüz sayısı ve karmaşıklığı
- Şartlı dallanma sayısı ve karmaşıklığı

- İç içe yerleşme seviyesi
- Veri yapıları türleri

Whitmire yaptığı çalışmada, karmaşıklık tanımlamasını geniş kapsamda incelemiş, karmaşıklık kavramına ilişkin topolojiye yer vermiştir. Çalışmasında hesaplama karmaşıklığı, psikolojik karmaşıklık, yapısal karmaşıklık ve programcı karmaşıklığı gibi kavramlara yer vermiştir. Hesaplama karmaşıklığını yazılımın çalıştırılabilmesi için gereksinim duyulan donanım (işlemci hızı, bellek ve disk kapasitesi vb.) cinsinden tanımlamıştır. Psikolojik karmaşıklık, yazılım tarafından çözülen karmaşıklık problemini ifade eder. Programcı karmaşıklığı ise programcının bilgi ve tecrübesine dayalı bir kavramdır [47].

Bazı işlevsellik ölçümleri, gereksinimler üzerinden tarif edilen problemlerin karmaşıklığını düzeltmeye çalışmaktadırlar. Albrecht'in teknik karmaşıklık faktörü tam da bu sorunun çözümü için önerilmiştir. Ancak problemlerin birden fazla çözümü olabilmesi ve bu çözümlere ait yaklaşımların değişkenlik göstermesi nedeniyle ölçümlenen karmaşıklık farklılaşabilecektir [27].

Fenton ve Pfleeger'e göre problem ve çözüm karmaşıklıkları birbirlerinden bağımsız olarak ele alınmalıdır. Problemlerin karmaşıklığı; problemlerin en iyi çözümü için gereken kaynak miktarı, çözümlerin karmaşıklığı ise belirli bir çözüm için gereken kaynak miktarı şeklinde tanımlanabilmektedir. Çözüm karmaşıklıkları iki farklı bakış açısı ile incelenebilir:

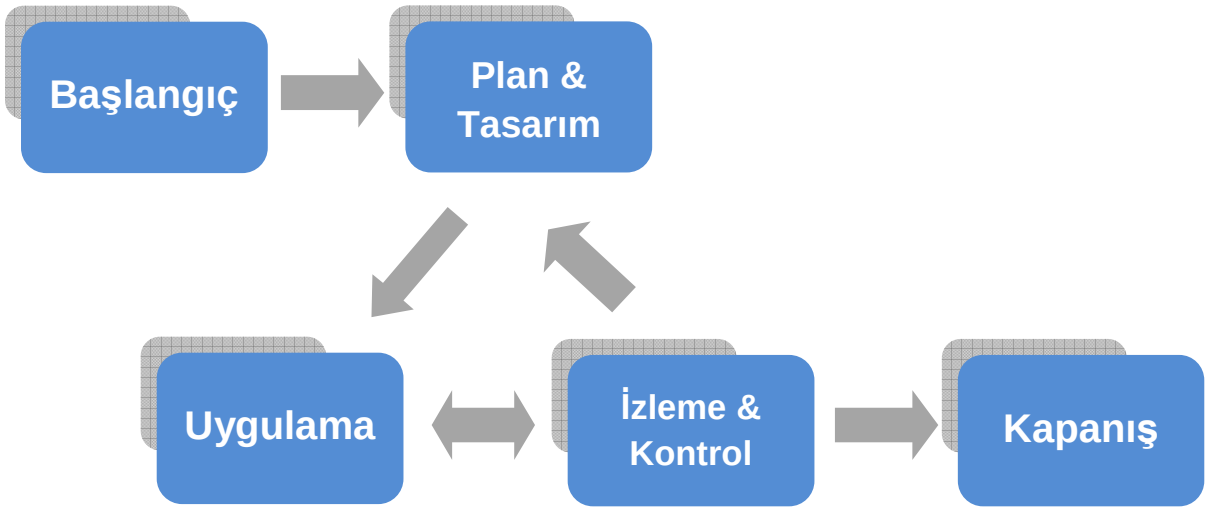
- Zaman ("time") karmaşıklığı: Sistem zamanı kaynak olarak kullanılır.
- Alan ("space") karmaşıklığı: Sistem hafızası kaynak olarak kullanılır.

2.3. İşlev Puanın Proje Yönetimindeki Rolü

Fenton ve Pflegger proje yönetimi tanımını, “paydaşların projeden beklentilerini karşılamak ya da aşmak için, proje etkinliklerine bilgi birikimi, beceri, araç ve tekniklerin uygulanması” olarak yapmışlardır [27]. Proje yönetimi konusunda genel yaklaşım aşağıdaki adımları içermektedir:

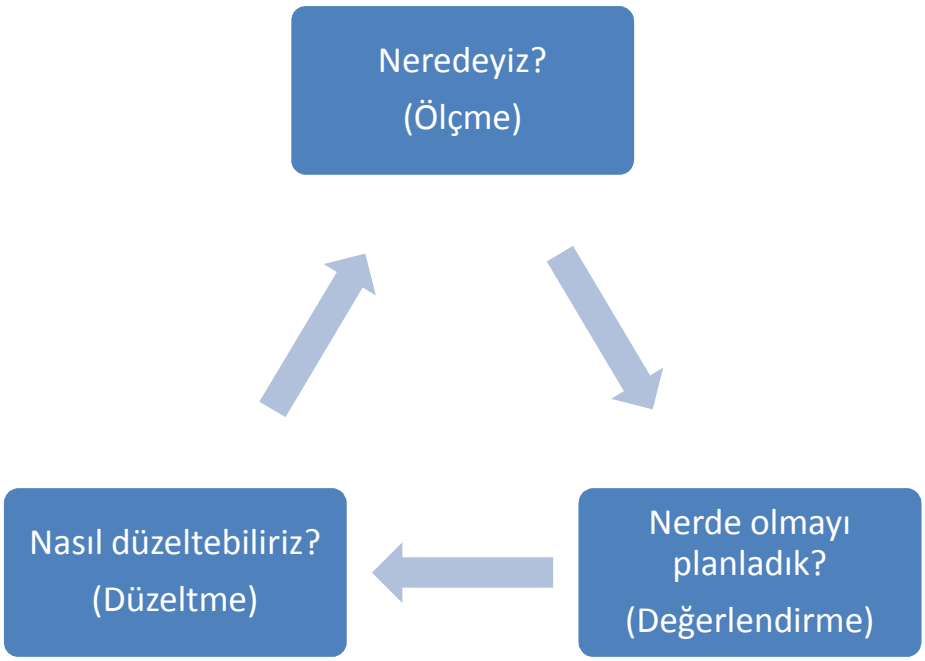
- Başlama
- Planlama ve Tasarım
- Uygulama
- İzleme ve Kontrol
- Kapanış

Proje yönetimi yaşam döngüsünde yukarıdaki adımlar Şekil 2.4’te gösterilen döngü içerisinde gerçekleşir.



Şekil 2.4 Proje Yönetimi Yaşam Döngüsü [48]

Ölçme, proje yönetimi adımları içerisinde izleme ve kontrol adımında ön plana çıkmaktadır. Proje yönetimi sürecinde ölçme, devam eden proje etkinlikleri üzerinden, projenin cari durumunda hangi konumda olunduğunun belirlenmesi için gerçekleştirilir. Daha sonra proje yönetim planına göre nerede olunması gerektiği belirlenir (iş gücü, maliyet, kapsam. vb açılarından) ve bu değerlendirmeye göre düzeltici aksiyonlar alınarak proje hedeflenen noktaya doğru yönlendirilir.



Şekil 2.5 Proje Değerlendirme Döngüsü [48]

İşlev puan metriği proje yönetiminde kullanılan anahtar metriklerden biridir. Birçok yazılım proje yöneticisi, ölçmenin proje geliştirme sürecini "anlamak" ve "yönlendirmek" konusunda kendilerine yardımcı olduğunu kabul etmektedir [27]. İşlev puan hesaplanması sayesinde projeye ilişkin değerlendirmeler yapılabilir, gelecek projeler için yapılması planlanan karşılaştırmalı yorumlar için taban oluşturulabilir. İşlev puan metriği kullanılarak;

- Proje için yapılan çalışmaların ne kadar verimli olduğu,
- Projenin maliyeti,
- Proje için harcanan iş gücü miktarı,
- Projenin hangi aşamada olduğu,

hesaplanarak değerlendirmeler yapılabilir.

2.3.1. Proje Yönetimi Süreci

Ölçme, yazılım projelerinin her adımında farklı şekillerde uygulanabilir. Proje henüz geliştirme aşamasına gelmeden planlama aşamasında, analiz, tasarım ve gereksinim belgeleri gibi bileşenler üzerinden; proje sürecinde harcanan işgücü, geçen süre, ortaya çıkan hatalar ve ortaya konan faaliyetler üzerinden; ürün üzerinde ise yazılım büyüklüğü, kalite ve teknik özellikler üzerinden gerçekleştirilebilir. Ölçme işleminin proje sürecindeki her adımda gerçekleştirilebilir oluşu, proje yönetimi sürecinin

başından sonuna kadar ölçme işleminden bağımsız olarak düşünülemeyeceğini göstermektedir.

Proje planlama aşamasında işlev puan aşağıdaki amaçlar için kullanılabilir.

1) Proje kapsamının belirlenmesi: Proje kapsamı, proje alt parçalara ayrılarak belirlenir.
2) Gereksinimlerin belirlenmesi ve değerlendirilmesi: Uygulama içerisindeki gereksinimlerin işlevsel olarak sayılması yolu ile elde edilir.
3) Risk değerlendirmesi: Planlanan proje büyüklüklerine göre risk değerlendirmesi yapılır. Yazılım büyüklükleri arttıkça yazılım bileşenleri arasındaki bağımlılık artacaktır. Çok büyük yazılım projeleri için karmaşıklık daha küçük parçalara ayrıştırma yolu ile kontrol edilememeye başlayacaktır [49].

Proje geliştirme aşamasında işlev puan aşağıdaki amaçlar için kullanılabilir.

1) Projenin işlevsel gelişiminin izlenmesi: Proje yönetimi sürecinin her adımında işlev puan analizi yapmak, işlevsel gelişimin izlenmesi için objektif bir kaynak sağlayacaktır.
2) Gerekli düzeltmelerin yapılması: Projenin işlevsel durumu, harcanan kaynakların durumu vb. verileri kullanılarak proje istenilen hedefe doğru yönlendirilir.

Proje geliştirimi sonrasında işlev puan aşağıdaki amaçlar için kullanılabilir:

1) Bakım maliyetlerinin planlanması: Uygulamanın kullanılması sırasında bakım için gerekli olabilecek iş gücü ve maliyetler projenin toplam işlevsel büyüklüğüne göre belirlenebilir.
2) Önceki uygulamalarla kıyas yapılması: Projenin bütünü ya da bazı modülleri işlevsel büyüklük değerleri üzerinden geçmiş proje verileri karşılaştırılabilir. Bu karşılaştırma sonucunda, proje yönetim süreçlerine ilişkin düzeltici aksiyonlar alınabilir. Her bir proje için verimlilik, maliyet ve işgücü verileri işlev puan metriği üzerinden değerlendirilebilir.

Ishigaki ve Jones yaptıkları çalışmada, ölçmenin proje yöneticilerine kazandıracağı yararları şu şekilde sıralamışlardır [50]:

- Etkin İletişim: Ölçme kurumun her seviyesindeki paydaşlar arasındaki iletişimi güçlü tutar. Objektif yapılan ölçümler belirsizlikleri ortadan kaldıracaktır.
- Problemleri erken belirleme ve düzeltme: Ölçme, pro-aktif bir yönetim sağlar. Potansiyel problemleri önceden belirlemek ve düzeltici aksiyonları almak, problem gerçekleştikten sonra alınacak aksiyonlara göre daha az maliyetlidir. İyi bir yönetici ortaya çıkabilecek bir problemi öngörerek gerçekleşmeden önce aksiyon alır.
- Yapılacak seçimlerde alternatif maliyetlerden haberdar olunur: Bir alanda yapılacak seçimler genelde diğer bir alanı da etkileyecektir. Ölçme bu etkiyi belirlemeye yardımcı olur böylelikle proje hedeflerine giden yolda en iyi kararların alınmasını sağlar.
- Belirli proje hedeflerinin izlenmesini sağlar: Projenin takvime uygunluğu, kalite gereksinimlerini karşılayıp karşılamadığı ve teslime hazır olup olmadığı güncel değerler ile proje planındaki değerler karşılaştırılarak anlaşılabilir.
- Risklerin Yönetilmesi: Riskleri etkin şekilde yönetebilmek için, tüm olası riskler ortaya çıkmadan önce ele alınmalı ve gerekli aksiyonlar alınmalıdır. Ölçme ve izleme sayesinde riskler erken evrelerde ele alınabilmektedir.
- Verilen kararları savunabilme ve haklı çıkarabilme: Proje yöneticileri kararlarını etkin bir biçimde almalıdırlar. Yöneticilerin birçoğu verdikleri karara temel olarak öznel kavramlar yerine nesnel kavramlar kullanmayı tercih ederler. Ölçme sayesinde projenin anlık perspektifi çizilebilir, bu sayede yöneticiler ölçme verilerini kullanarak uygun aksiyonları alabilir.

2.3.2. Kazanılan Değer (“Earned Value - EV”)

Kazanılan Değer, 20. Yüzyılın üçüncü çeyreğinde Amerika Birleşik Devletleri hükümeti yönetim programlarında finansal analize ilişkin bir kavram olarak ortaya çıkmış ve aynı yüzyılın dördüncü çeyreği ile birlikte proje yönetimi alanında önemi giderek artmış bir kavramdır [51]. Aynı yıllarda Amerikan Savunma Bakanlığı’nın bu kavramı proje performanslarını ölçmeye yönelik standart bir metot olarak benimsemesiyle birlikte kullanımı yaygınlaşmaya başlamıştır [52]. Bu yöntem, gerekli iş gücü ve maliyet açısından bazı kesimler tarafından desteklenmemektedir. Bu kişilerce yöntemin uygulanmasıyla elde edilen başarımın sınırlı olduğu savunulmaktadır. Kazanılan Değer kavramını destekleyenler ise maliyet kazanımları, gelişmiş analiz gerçekleştirimleri, iletişim ve kontroller üzerinde durmaktadır.

2.3.2.1. Kazanılan Değer Kavramı Nedir?

Kazanılan Değer kavramına ilişkin birçok tanım yapılmıştır. Özünde birbirine paralel olan bu tanımlar, farklı bakış açılarına sahip olmaları nedeniyle birbirlerinden ayrılmaktadırlar.

Kazanılan Değer,

- Microsoft Project 2003 kullanıcı dokümanında, "proje performansını ölçmek için kullanılan bir metot" şeklinde tanımlanmıştır. Devamında ise "O ana kadar yapılan iş miktarı, görev ve kaynak taban maliyetleri ışığında ne ölçüde bir bütçenin harcanmış olması gerektiğini gösterir." cümlesi ile söz konusu tanım bir derece daha detaylandırılmıştır [53].
- NASA tarafından, "Anlaşmalı taraf veya faaliyet alanının program maliyetleri ile başarımını değerlendirme, anlama ve ölçme için tanımlı bütünleşik yönetim kontrol sistemi" şeklinde tanımlanmıştır [54].
- Amerika Birleşik Devletleri Savunma Bakanlığı tarafından, "Yapılan sözleşmelerdeki teknik, maliyet ve zaman parametrelerini birleştiren bir program yönetim aracı" şeklinde tanımlanmıştır [55].

Daha genel bir anlatımla Kazanılan Değer, proje planını, mevcut iş durumunu ve tamamlanan iş değerlerini izleyerek projenin plana uygun olarak ilerleyip ilerlemediğini takip edebilmek için kullanılan bir yaklaşımdır. İsminin de çağrıştırdığı gibi, harcanan iş gücü üzerinden sağlanan bir kazanımdır. Proje yönetiminde bu kazanım, aktivitelerin sona ermesiyle elde edilebilmekte ve aynı zamanda Kazanılan Değer ilerlemenin bir metriği olarak ele alınabilmektedir.

Tamamlanan iş ile Kazanılan Değer arasında doğrudan bir ilişki bulunmaktadır. Bu noktada projenin işlevsel olarak büyüklüğü ile harcanması gereken iş gücü arasında doğru orantı olması nedeniyle işlevsel büyüklük – işgücü ilişkisi karşımıza çıkmaktadır. Projenin herhangi bir anında işlevsel olarak tamamlanan kısım Kazanılan Değer değerini karşılamaktadır. Bu noktadan hareketle projenin cari durumu değerlendirilebilmekte, geliştirme sürecine ilişkin çıkarımlar yapılabilmektedir.

Kazanılan Değer,

- projenin tümü ya da herhangi bir alt bölümü için ilerlemenin ölçüm birimidir.

- projenin ilerlemesi ve performansının analizi için tutarlı bir metottur.
- projenin maliyet performansının analizi için temel oluşturmaktadır [52].

Proje yöneticisi Kazanılan Değer üzerinden kritik etmenleri belirleyebilir, ileriki maliyet ve zaman performanslarını kestirebilir ve proje planından saplamalar olması durumunda düzeltici aksiyonlar alarak projenin yeniden rayına oturmasını sağlayabilir. Bu sebeple Kazanılan Değer Yönetimi ("Earned Value Management - EVM)" hem performans ölçümünü hem de performans yönetimini kapsamına almaktadır. Ayrıca, Kazanılan Değer kavramı, proje sözleşmelerinde her iki taraf için de büyük katkılar sağlamaktadır [55].

EVM çeşitli kullanım biçimleriyle performans ölçümü için geniş kullanımı olan bir yöntemdir. Projenin kapsam, maliyet ve takvim bilgilerini birleştirerek, proje yönetimi ekibine proje performansı ve ilerlemesini değerlendirebilme ve ölçme hususlarında yardımcı olmaktadır. EVM farklı endüstri alanlarındaki tüm projeler için uygulanabilir niteliktedir [56].

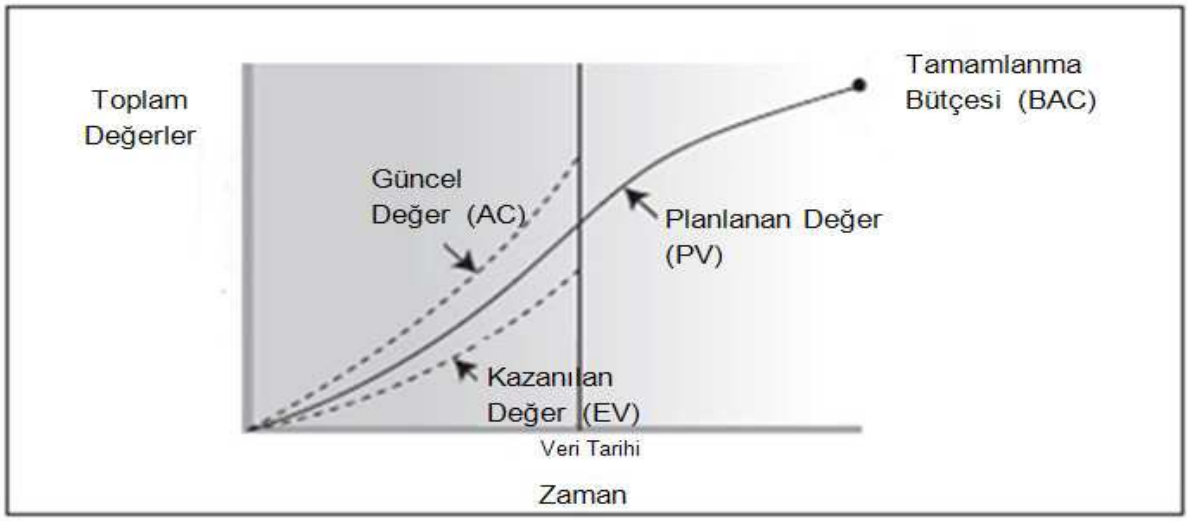
Kazanılan Değer bir proje için planlanan, gerçek ve bütçelenen değerler arasında karşılaştırma yaparak, projenin cari durumu ile ilgili değerlendirme yapma fırsatı vermektedir [57]. Kazanılan Değer değerlendirmesi yapılırken birçok unsur kullanılmaktadır. Kazanılan Değer kavramını anlayabilmek için, bu unsurların ne olduklarını bilmek gerekir. EVM, her bir projeyi üç anahtar bileşen bazında değerlendirmekte ve izlemektedir (Şekil 2.6) [56].

Planlanan Değer ("Planned Value – PV"): Planlanan değer; bir aktivite veya iş ayrıştırma yapısı ("work breakdown structure – WBS") bileşeni için gerçekleştirilecek işlere atanan yetkilendirilmiş bütçedir. Planlanan değerin bütünü Performans Ölçüm Tabanı ("Performance Measurement Baseline – PMB") olarak isimlendirilir. Proje için toplam planlanan değer (ayrılmış bütçe) ise Tamamlanma Bütçesi ("Budget At Completion - BAC") olarak ele alınmaktadır.

Kazanılan Değer ("Earned Value – EV"): Kazanılan değer belirli bir iş veya iş ayrıştırma yapısı (WBS) bileşeni için ayrılmış bütçe cinsinden tamamlanan iş değerini ifade etmektedir. Tamamlanmış iş adımları ve mevcut iş adımı için bütçeyi, diğer bir deyişle iş ilerlemesinin anlık değerini göstermektedir Kazanılan değer genellikle projenin tamamlanma yüzdesini tanımlamaktadır. Bu nedenle kazanılan değer metriği PMB ile ilişkili olmalıdır ve kazanılan değer kesinlikle, o bileşen için planlanan

değerden büyük olmamalıdır. İş adımı ölçüm kriteri, devam eden işi ölçebilmek amacıyla her bir iş ayrıştırma yapısı (WBS) bileşeni için uygulanmalıdır. Proje yöneticisi EV değerini, cari durumu görebilmek için artırımsal olarak ve uzun dönem performans değerlendirme yapabilmek için kümülatif olarak izlemektedir.

Gerçek Maliyet ("Actual Cost – AC"): Gerçek maliyet, bir aktivite veya iş ayrıştırma yapısı (WBS) bileşeni için sonuçlandırılmış işlere ait gerçekte katlanılan ve kaydedilen maliyettir. Diğer bir deyişle, kazanılan değer için katlanılan toplam maliyettir. AC değerinin, PV ile bütçelenmiş ve EV ile ölçülen değerlerle uyumlu olması gerekmektedir. Bir üst limiti olmamakla beraber, ölçülecek EV için harcanan değer olarak ifade edilmektedir.



Şekil 2.6 Kazanılan Değer, Planlanan Değer ve Gerçek Değer

Kazanılan Değer Yönetiminde, kabul edilen plandan sapmalar da izlenebilmektedir:

Zamanlama Sapması ("Schedule Variance - SV"): Zamanlama sapması, projenin zamanlama performansının bir metriğidir ve kazanılan değer ve planlanan değerin farkına eşittir. Zamanlama sapması, planın gerisine düşülüp düşülmediğini gösteren önemli bir metriktir. Proje tamamlandığı zaman planlanan değerlerin tamamının kazanılmış olması nedeniyle, zamanlama sapması sıfır değerine ulaşacaktır. Zamanlama sapmaları, kritik yol metodolojisi ("critical path methodology") ve risk yönetimi ile birlikte kullanılırlar.

$$SV = EV - PV$$

Maliyet Sapması ("Cost Variance - CV"): Maliyet sapması, projenin maliyet performansını göstermektedir. Maliyet sapmasının, kazanılan değer ile gerçek

maliyetin farkına eşit olması nedeniyle söz konusu değer proje sonunda maliyet sapması, tamamlama maliyeti (BAC) ile gerçek harcanan miktarın farkına eşit olacaktır. Maliyet sapması, fiziksel performans ile harcanan maliyetlerin ilişkisini göstermesi bakımından kritiktir, bu nedenle maliyet sapmasının negatif olması çoğunlukla telafisi olmayan projeleri ifade etmektedir.

$$CV = EV - AC$$

Maliyet ve zamanlama sapması değerleri, projelerin maliyet ve zamanlama performansları açısından karşılaştırılabilmesi için etkin metrikler olarak kullanılabilirler. Sapma ve metrikler projenin durumunu belirleyebilmek ve proje maliyeti ve zamanlaması için plan oluşturmak adına kullanılabilmektedir.

Zamanlama Performans Göstergesi ("Schedule Performance Index - SPI"): Zamanlama performans göstergesi, projeler için sağlanan ilerleme ile planlanan ilerlemenin karşılaştırılabilmesi, aynı zamanda nihai proje tamamlama tarihlerinin planlanabilmesi amacıyla maliyet performans göstergesi (CPI) ile birlikte kullanılır. Zamanlama performans göstergesi değeri 1.0 altında ise planlanandan daha az işin tamamlandığı, 1.0 üzerinde ise planlanandan daha fazla işin tamamlandığı anlaşılmaktadır. Zamanlama performans göstergesinin, proje boyunca tamamlanan bütün işi ifade etmesi nedeniyle projenin tamamlanma sürecinin plana göre geride olup olmadığının anlaşılması adına kritik öneme sahiptir ve analiz edilmesi gerekmektedir. Zamanlama performans göstergesi değeri, kazanılan değerin, planlanan değere oranıdır.

$$SPI = EV/PV$$

Maliyet Performans Göstergesi ("Cost Performance Index - CPI"): Maliyet performans göstergesi değeri tamamlanan iş değerinin projede yapılan gerçek maliyet veya ilerleme ile karşılaştırılması sonucu ortaya çıkan bir değerdir. Tamamlanan işin maliyet performansını gösteren, EVM için en kritik göstergedir. CPI değerinin 1.0 den küçük olması, maliyetlerin tamamlanan işin üzerinde olduğunu, büyük olması ise maliyetlerin tamamlanan işin altında kaldığını göstermektedir. Maliyet performans göstergesi değeri, kazanılan değerinin gerçek maliyete oranı şeklinde ifade edilir.

$$CPI = EV/AC$$

2.3.2.2. Neden Kazanılan Değer?

Kazanılan Değer metodunun kullanımı öncesinde kullanım amaçları belirlenmelidir. Yukarıda yer alan tanımlardan da hatırlayacağımız üzere, Kazanılan Değer ortak bir metrik, tutarlı bir metodoloji ve performans analizi için temel oluşturmaktadır.

Bir proje yöneticisi, en başından en sonuna kadar projenin durumunu analiz etmek ve buna uygun kararlar almak zorundadır. Bir proje birbirinden farklı nitelikte birçok alt adımdan oluşmakta ve bu adımlarda ilerlemenin maliyeti farklılaşmaktadır. Örneğin bir satır analiz raporu yazmak ile bir satır kod yazmanın maliyetleri birbirinden farklı olacaktır. Ancak projenin başında bu alt adımların her biri için ortak birime sahip bir planlama yapılması durumunda ilerleme proje yöneticisi tarafından çok daha rahat bir şekilde izlenebilecektir. Örneğin her alt adım için planlanan çalışma süreleri belirlendiğinde, projenin mevcut durumunda hangi noktada olunduğuna dair fikir elde edilebilmektedir. Dolayısıyla, Kazanılan Değer bize çok farklı iş gereksinimleri için ortak bir ilerleme değerlendirmesi yapma imkanı sağlamaktadır. Örneğin bankacılık sektörü için Kazanılan Değer, ürün geliştirimi ile market araştırmaları, sistem tasarımı, pazarlama ve ürün tanıtımını birleştirmeyi sağlamaktadır. [52]

Ayrıca Kazanılan Değer çok farklı disiplinlerde gerçekleştirilen bütün projeler için kullanılabilmektedir, bu da farklı disiplinler için proje izleme ve değerlendirme noktasında ortak bir standardın olmasını sağlamaktadır. Ayrıca Kazanılan Değer kullanımı projenin maliyet analizi performansını arttırmaktadır. Geleneksel yöntemler tamamlanan iş için gerçek maliyetler üzerinde dururken, Kazanılan Değer yapılan iş için tutarlı ve maliyet açısından karşılaştırılabilir ölçme imkanı sağlamaktadır [52].

Kazanılan Değer Yönetimi (EVM) için kurulum ve kullanımda bazı adımları gerçekleştirmek gereklidir.[52] Kurulum aşaması için;

- Projeyi yönetilebilir parçalara ayırmak için WBS (Work Breakdown Structure) uygulanması,
- Zaman planı yapmak için projenin bütününü temsil eden aktivitelerin belirlenmesi,
- Her aktivite için sarf edilecek maliyetlerin belirlenmesi,
- Aktivitelerin zamana göre planlanması,
- Planın kabul edilebilirliğini teyit edebilmek için verilerin katalog haline getirilmesi

adımlarının tamamlanması gerekmektedir.

Kullanım aşaması için ise,

- Zamanlamanın, aktivite ilerlemeleri raporlanarak güncellenmesi,
- Aktiviteler üzerindeki gerçek maliyetlerin girilmesi,
- Kazanılan Değer hesaplamalarının çalıştırılması, rapor ve planların yazdırılması ve düzenlenmesi,
- Verilerin analiz edilmesi ve performans raporunun hazırlanması,

adımlarının tamamlanması gerekmektedir.

Sonuç olarakproje yönetiminde yeni bir konu olmamasına karşın, EVM kavramının popülaritesi giderek artmakta ve projenin izlenebilmesi, değerlendirilmesi ve gerekli aksiyonların alınması noktasında bu kavram, kritik öneme sahip olmaktadır.

2.3.2.3. Measurement Kütüphanesi'nin Kazanılan Değer Analizi için Kullanılması

Kazanılan Değer tüm disiplinlerde kullanılabilen bir yöntemdir. Yazılım projelerinde de hedeflenen ve cari durumu karşılaştırmak, analiz etmek ve buna göre gerekli aksiyonları almak için kullanılabilmektedir.

Measurement Kütüphanesi'nin işlevsel yazılım büyüklüğü ölçümünün otomatikleştirilmesi için geliştirilmiş bir kütüphane olduğu ve ölçme işlemini gerçekleştirebilmesi için kod içine bazı eklemelerin yapılması gerektiği konusu giriş bölümünde vurgulanmıştı. Bu eklemeler, projenin henüz geliştirme aşamalarında yapılabileceği gibi, geliştirimi tamamlanmış bir proje üzerinde de yapılabilir. Measurement Kütüphanesi'nin proje geliştirme aşamalarına entegre olarak kullanılmasıyla, projenin işlevsellik anlamında güncel durumu izlenebilir, hedeflenen ve cari durum karşılaştırılabilir. Projenin cari durumunda planlanan değer (PV), kazanılan değer (EV) ve gerçek maliyet (AC) değerleri hesaplanarak değerlendirmeler yapılabilir. Proje tamamlandığında hedeflenen maliyet (tamamlama maliyeti – BAC) ile gerçek maliyet karşılaştırmaları yapılabilir.

Örneğin, bir yazılım projesinin 20 alt işlevden oluştuğunu varsayalım. Her bir alt işlev için, işlevsel kullanıcı gereksinimleri yardımıyla yapılan tahminler ile hedeflenen

işlevsel büyüklükler ve zamanlamalarının tek tek belirlenmiş olduğunu, proje için belirlenen bitiş süresi ve bütçe değerlerinin aşağıdaki tabloda gösterildiği gibi olduğunu düşünelim ve farklı senaryolar için durumları inceleyelim.

Çizelge 2.5 Varsayılan proje dahilindeki alt işlevler ve planlanan büyüklükler

Zamanlama (Hafta no)	Alt işlevler	Planlanan İşlevsel Büyüklük	Planlanan Süre (adam-saat)	Toplam Proje Bütçesi (TL)
1	Aİ1	6	380	38000
1	Aİ2	4		
1	Aİ3	8		
2	Aİ4	8		
2	Aİ5	7		
2	Aİ6	9		
2	Aİ7	12		
3	Aİ8	9		
3	Aİ9	14		
3	Aİ10	14		
4	Aİ11	9		
4	Aİ12	8		
4	Aİ13	4		
4	Aİ14	6		
4	Aİ15	3		
5	Aİ16	8		
5	Aİ17	6		
5	Aİ18	4		
5	Aİ19	8		
5	Aİ20	5		
	TOTAL	152		

Senaryo1: Projenin herhangi bir anında, Measurement Kütüphanesi sayesinde, tam anlamıyla çalışan işlevler için anlık işlevsel büyüklük hesaplanabilecektir. Örneğin, projenin ikinci haftasının sonunda alt işlevlerin ilk beş tanesinin gerçekleştiriminin yapıldığını ve söz konusu anda cari işlevsel büyüklüğün 33 çıktığını varsayalım. Bu noktada EV değeri (ikinci haftanın sonunda tamamlanan işlevsel büyüklük) 33 olacaktır. PV değeri (ikinci haftanın sonunda planlanan işlevsel büyüklük) ise 54'tür. Bu noktada proje yöneticisi, ilgili sapmanın sebebine odaklanabilecek ve gerekli aksiyonları alabilecektir.

Senaryo2: Benzer şekilde, ölçümün yapıldığı anda işlevsel büyüklüğün planlandığı gibi ikinci hafta sonunda 54 olarak bulunduğunu varsayalım. Projenin işlevsellik anlamında hangi yüzdesinin tamamlandığı bir diğer deyişle projenin ilerlemesinin

hangi noktada olduğunu planlanan toplam işlevselliğin, tamamlanan işlevselliğe oranlanarak bulunabilecektir. Bu örnekte planlanan toplam işlevsellik değeri 152 olduğundan, projenin işlevsellik anlamda tamamlanma oranının 54 / 152 = %35,5 seviyelerinde olduğu görülmektedir. Proje yöneticisi hedeflenen ve tamamlanan işlevsellik ile birlikte hedeflenen ve kullanılan zaman değerleri ışığında projenin hedeflenen şekilde tamamlanabilmesi için yapılması gerekenler ile ilgili karar ve aksiyonları alabileceklerdir.

Senaryo3: İşlevsel büyüklük hesaba katılmadan yalnız proje bütçesi üzerinden değerlendirmelerin yapıdığı durumda, üçüncü hafta sonunda planlandığı gibi 91 işlev puanlık işin tamamlandığı durumda, proje bütçesinden 28000 TL harcandığını düşünelim. Eğer maliyet üzerinden değerlendirme yapıyor olsaydık, harcanan bütçe kısmını toplam bütçeye oranlayarak projenin hangi oranda tamamlandığını bulmaya çalışacaktık. Bu durumda projenin 28000/38000 = %73 oranında tamamlandığı hesaplanacaktı. Ancak işlev puan üzerinden gidildiğinde, uygulamadan beklenen işlevselliğin ancak 91/152 = %60 oranında karşılanabildiği hesaplanmaktadır. Bu durum projenin maliyetlerini aştığını göstermektedir. Bu noktada işlevsel büyüklüğün otomatik olarak çok kısa bir süre içerisinde hesaplanabilmesi projenin cari durumunun değerlendirilebilmesi ve gerekli aksiyonların alınabilmesi için oldukça kritiktir.

Senaryo4: Senaryo 3’e benzer şekilde harcanan işgücü üzerinden projenin geldiği noktayı hesaplamak da doğru sonucu vermeyecektir. Örneğin üçüncü hafta sonunda projenin 91 işlev puanlık kısmının tamamlandığı ve 170 adam-saat harcandığı bir durumda, harcanan iş gücü üzerinden yapılan hesap ile projenin 170/380 = %44 oranında tamamlandığını düşünebiliriz. Ancak işlev puan üzerinden yapılan hesapta, uygulamanın karşılaması gereken işlevselliğin cari durumda 91/152 = %60 oranında karşılandığını görebiliriz. Bu durum bize projede olunması gereken noktanın ilerisinde olunduğunu göstermektedir. Bu durumda proje yöneticisi tarafından gerçeğe yakın planlama için aksiyon alınabilir ve kaynakların gelecekte daha etkin kullanımı sağlanabilir.

İncelediğimiz senaryolarda, işgücü ve maliyetin tek başına projenin cari durumu için doğru bilgiler üretemeyeceği, işlevsel büyüklüğün kullanımı ile projenin gidişatı hakkında net yorumlar yapılabildiği görülmektedir. Bu durum işlevsel büyüklüğün

Kazanılan Değer ve dolayısı ile proje yönetimi açısından önemini bir kez daha göstermektedir.

3. COSMIC İşlevsel Büyüklük Ölçümü

Yazılımların işlevsel büyüklüklerinin ölçülmesi kavramı ilk kez 1979 tarihinde Allan Albrecht tarafından ortaya atılmıştır [5]. Çıkış noktası, kod satır sayısı metriğinin programlama dili ve platforma bağımlı olması ve yazılım büyüklük ölçümlerini bu bağımlılıktan kurtarmanın gerekliliği olmuştur.

Tez çalışmasında COSMIC metodu kullanılmıştır. Bu metodun seçilmesinde ücretsiz olması, ISO tarafından tanınan bir standart olması, kolay uygulanabilirliği, çok çeşitli yazılım uygulamalarında kullanılabilirliği, yazılımın büyüklüğünü birden fazla bakış açısından ölçebilmesi ve COSMIC kavramlarının modern yazılım mühendisliği kavramlarıyla örtüşmesi [24] etkili olmuştur.

COSMIC metodunun temeli ilk kez 1997 yılında, Albrecht FPA metodunun bazı noktalarda farklı bakış açılarıyla geliştirilmiş bir şekli olarak ortaya çıkmış, yapılan testlerde benzer uygulamalar için başarılı sonuçlar ürettiği görülmüştür. 1998'de FFP metodunun geliştiricileri, o dönem ikinci nesil işlevsel ölçüm metodunun prensiplerini tanımlayan COSMIC grubuyla birleşmiş ve 1999 yılında COSMIC-FFP v2.0 ilk kez ortaya çıkmıştır. V3.0 ile birlikte COSMIC olarak anılmaya başlamıştır [22].

COSMIC (Common Software Measurement International Consortium), tüm dünyadan gönüllü akademik ve uygulayıcı üyeleri olan ve amacı yazılım büyüklüğü ölçme konusunda yeni yaklaşımlar geliştirmek ve bu yaklaşımların uygulanabilirliklerini araştırmak ve doğrulamak olan bir gruptur.

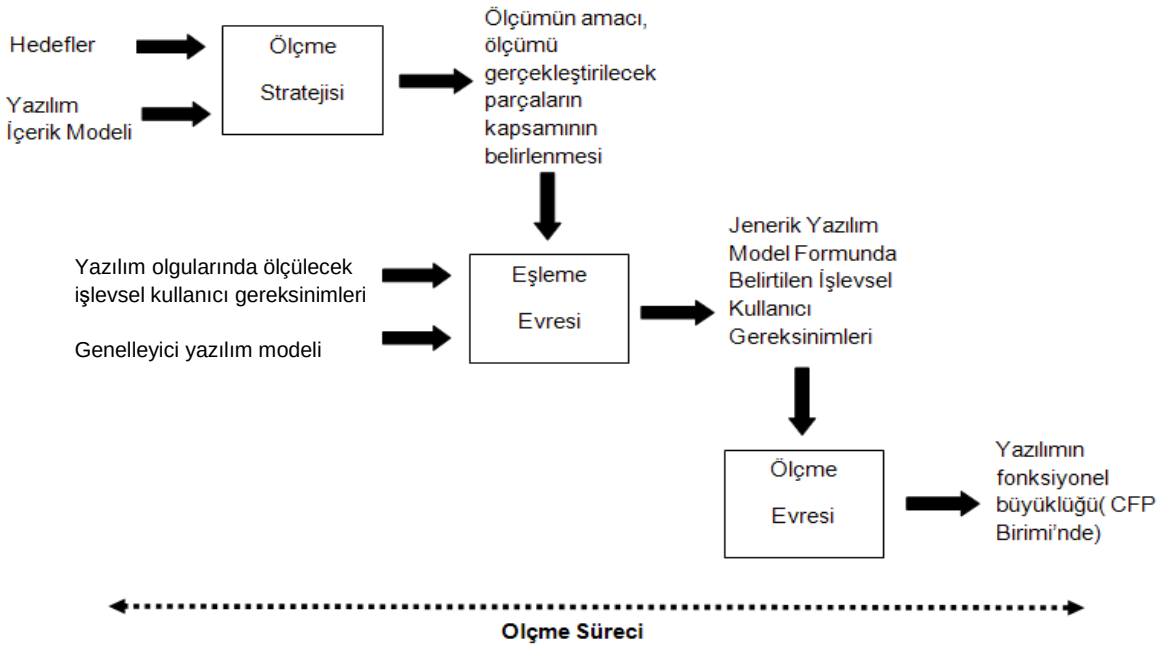
COSMIC metodunun kullanımı, bankacılık, sigortacılık gibi iş yönetimini destekleyici MIS yazılımları için uygundur. Ancak matematik karmaşıklığı yüksek olan, simülasyon ve kendiliğinden öğrenen yazılımlar için uygulanabilir değildir [7].

COSMIC ölçme yöntemi, yazılımların işlevselliğini ölçmeyi hedefler. İşlevsellik yazılımın kullanıcılar için gerçekleştirdiği bilgi işleme faaliyetlerinin tümüdür. İşlevsel kullanıcı gereksinimleri ("Functional User Requirements – FUR") ise işlevsel kullanıcılar için yazılımın ne yapması gerektiğini tanımlar. Bu gereksinimler herhangi bir teknik ya da kalite ihtiyacını dışlar. İşlevsel büyüklük ölçümü yapılırken sadece yazılımın işlevselliği dikkate alınır.

İşlevsel kullanıcı gereksinimleri geliştirme öncesinde ya da geliştirme işleminden sonra belirlenebilir. Geliştirme öncesinde, gereksinim tanımları ve analiz

dokümanlarından yapılan çıkarım ile belirlenebilen bu gereksinimler, yazılım geliştirildikten sonra fiziksel programlar, yazılım işlem ve prosedür kılavuzlarından ve veri tabanı işlemleri üzerinden belirlenmeye çalışılır.

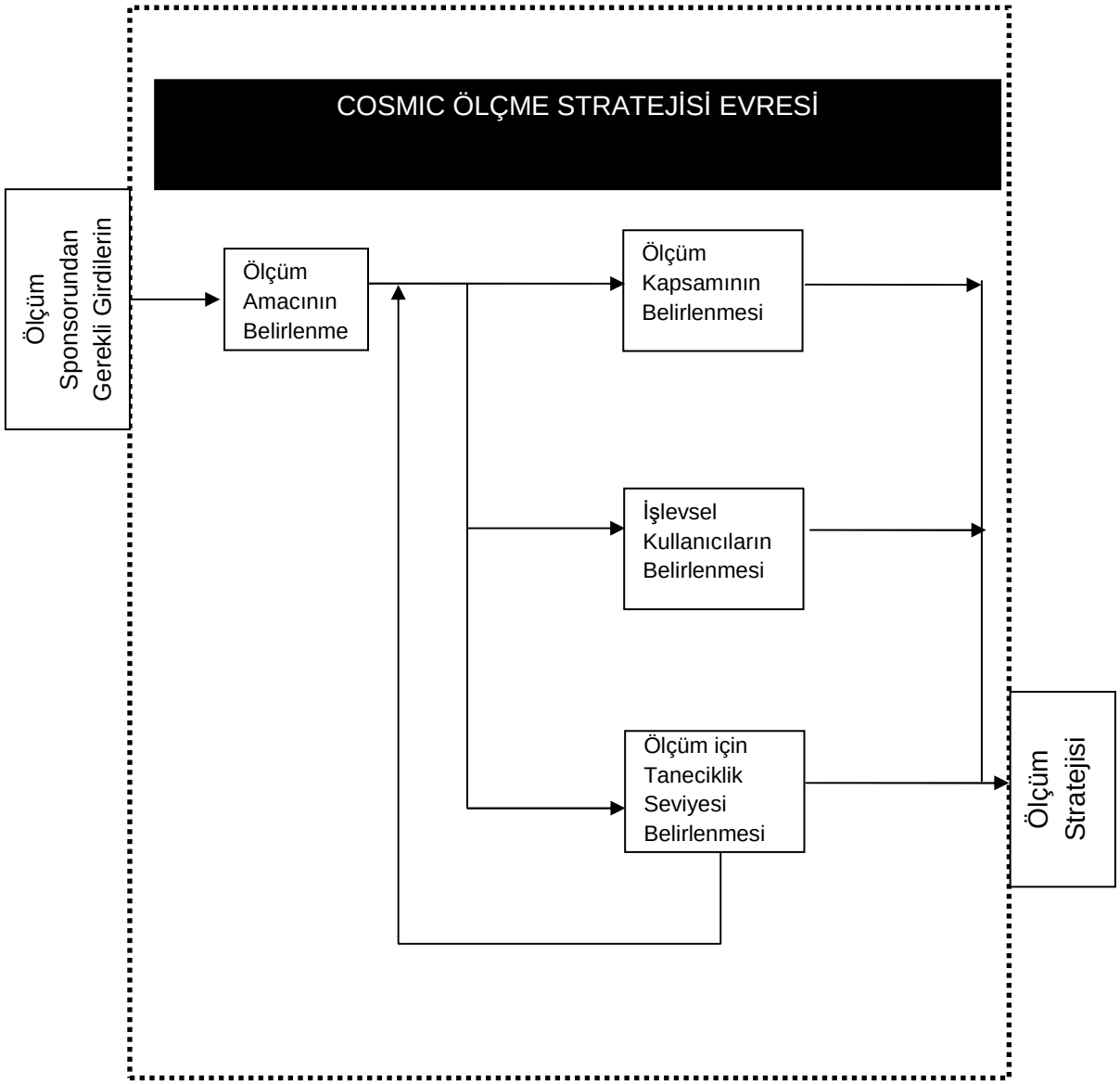
COSMIC Measurement Manual [7], ölçme için üç temel evre tanımlamaktadır (Şekil 3.1): Ölçme Stratejisi evresi, Eşleme evresi, Ölçme evresi.



Şekil 3.1 COSMIC Yöntemi Yapısı

3.1. Ölçme Stratejisi Evresi ("Measurement Strategy Phase")

Bu evrede; yapılacak ölçme işleminin amacı, kapsamı, ölçme işleminin gerçekleştirileceği uygulamanın işlevsel kullanıcıları ve ölçmenin gerçekleştirileceği taneciklik seviyesi belirlenir. Bu kavramlar ölçme sonrasında yapılacak karşılaştırma ve değerlendirmeler için de kullanılmaktadır. Ölçme evresine ilişkin akış şemasına Şekil 3.2'de yer verilmiştir.



Şekil 3.2 Ölçüm Stratejisinin Belirlenmesi Süreci [7]

Amaç, ölçme işleminin neden yapıldığını ve sonuçlarının ne için kullanılacağını belirtir. Ölçme amacının belirlenmesi kapsam ve işlevsel kullanıcıların belirlenmesi açısından önemlidir.

Kapsam, ölçmeye dahil edilecek işlevsel kullanıcı gereksinimlerini ifade eder. Kapsam içerisine birden fazla yazılım katmanı dahil olamaz. Katman; donanımla birlikte bütün bir bilgisayar sistemini oluşturan yazılım sisteminin işlevsel olarak parçalanmasıyla elde edilen birimdir. Hiyerarşik olarak organize olmuşlardır, hiyerarşide her seviyede bir adet katman bulunur. Bu kavram katmanlı yazılım mimarisi kavramı ile benzerlik göstermektedir. Bir katman kullanıcıları için işlevsel hizmetler sağlar. Alt katmandakiler üst katmandaki yazılımlar için hizmet sağlarlar. Ölçme kapsamında dış davranış ele alınıyor ise katmanların işlevselliği ele alınmaz. Bu noktada ölçümü yapılacak birim için kara kutu yaklaşımı sergilenir. Ancak katmanlar için ayrı ayrı ölçme gerçekleştirilmesi gereken durumlarda katmanların

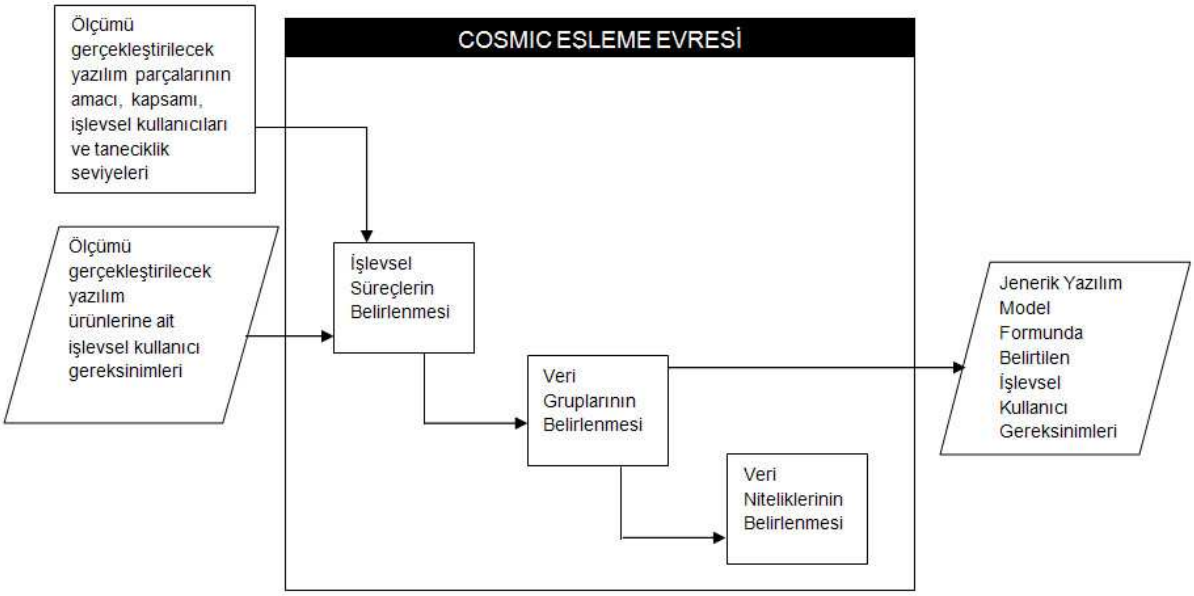
işlevselliği dikkate alınır. Bu durumda bir katman için “yazma” niteliği taşıyan bir hareket diğer katman için “giriş” niteliği taşıyabilecektir.

İşlevsel kullanıcılar; bir yazılım birimi için tanımlı işlevsel kullanıcı gereksinimleri içerisinde, veri alan ya da gönderen kullanıcıdır. İşlevsel büyüklük, işlevsel kullanıcıya göre değişiklik gösterir. Bunun sebebi gereksinimlerinin farklı olmasıdır. COSMIC ölçüm metodu farklı işlevsel kullanıcı bakış açılarıyla ölçüm yapabilmektedir. Bu sebeple ölçümün hangi işlevsel kullanıcılar temel alınarak yapılacağının belirlenmesi büyük önem arz etmektedir. Ölçümü yapılan yazılım ve onun işlevsel kullanıcıları arasındaki kavramsal arayüze “sınır” adı verilmektedir. Sınır, ölçümü yapılacak yazılımın işlevsel kullanıcıları açısından içeriğinin ne olduğunun ve neyin ölçüme dahil edildiğinin anlaşılmasını sağlar.

Taneciklik seviyesi, ölçme işleminin hangi ayrıntı seviyesinde yapılacağını gösterir. Yazılım geliştirmenin her evresinde yazılımın taneciklik seviyesi artacaktır. Taneciklik seviyesinin artması kamera ile yakınlaştırma işleminin yapılmasına benzetilebilir. Görüntü yakınlaştıkça daha fazla ayrıntı görülebilecektir. Yapılan ölçümlerin birbirleriyle karşılaştırılabilmeleri için ölçme, aynı taneciklik seviyesinde gerçekleştirilmiş olmalıdır.

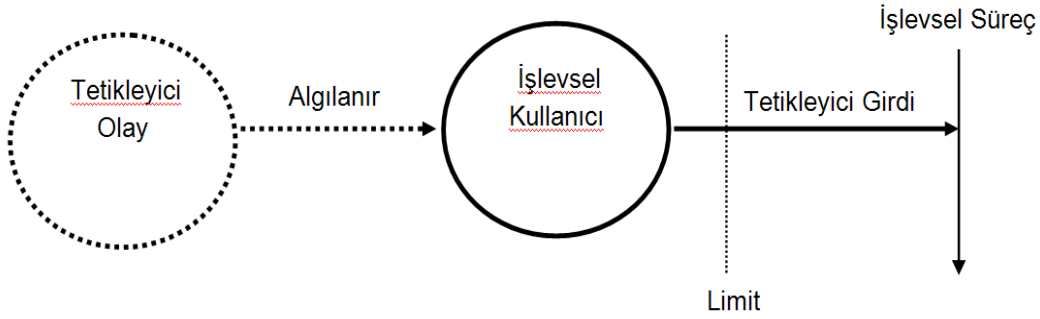
3.2. Eşleme Evresi (“Mapping Phase”)

Bu evrede belirlenen işlevsel kullanıcıların tetikledikleri olay kümeleri diğer bir deyişle işlevsel süreçler saptanır, sonrasında bu süreçlerle ilgili veri grupları ve veri özellikleri belirlenir. Eşleme evresine ilişkin gösterim Şekil 3.3’te verilmiştir.



Şekil 3.3 Eşleme Evresi

İşlevsel süreç keşfi; İşlevsel kullanıcı gereksinimleri üzerinden, yazılımın bir bölümü için işlevsel süreçlerin belirlenmesidir. İşlevsel süreç ise, işlevsel kullanıcı gereksinimleri kümesinin her biri bağımlı ya da bağımsız çalıştırılabilen veri hareketleri kümesidir. Bir ya da daha fazla işlevsel sürecin başlamasını sağlayan olaya "Tetikleyici olay" adı verilmektedir. Tetikleyici olay, işlevsel süreç ve işlevsel kullanıcı arasındaki ilişki Şekil 3.4'te verilmiştir. COSMIC FFP kapsamında bir işlevsel süreç bir "Giriş" ve onu takip eden "Çıkış" veya "Yazma" veri hareketleri olmak üzere en az iki veri hareketinden oluşmaktadır. İşlevsel süreç, tetiklenen olay sonucu dönen cevap ile birlikte, bekleme moduna dönülmesiyle birlikte sonlanır.

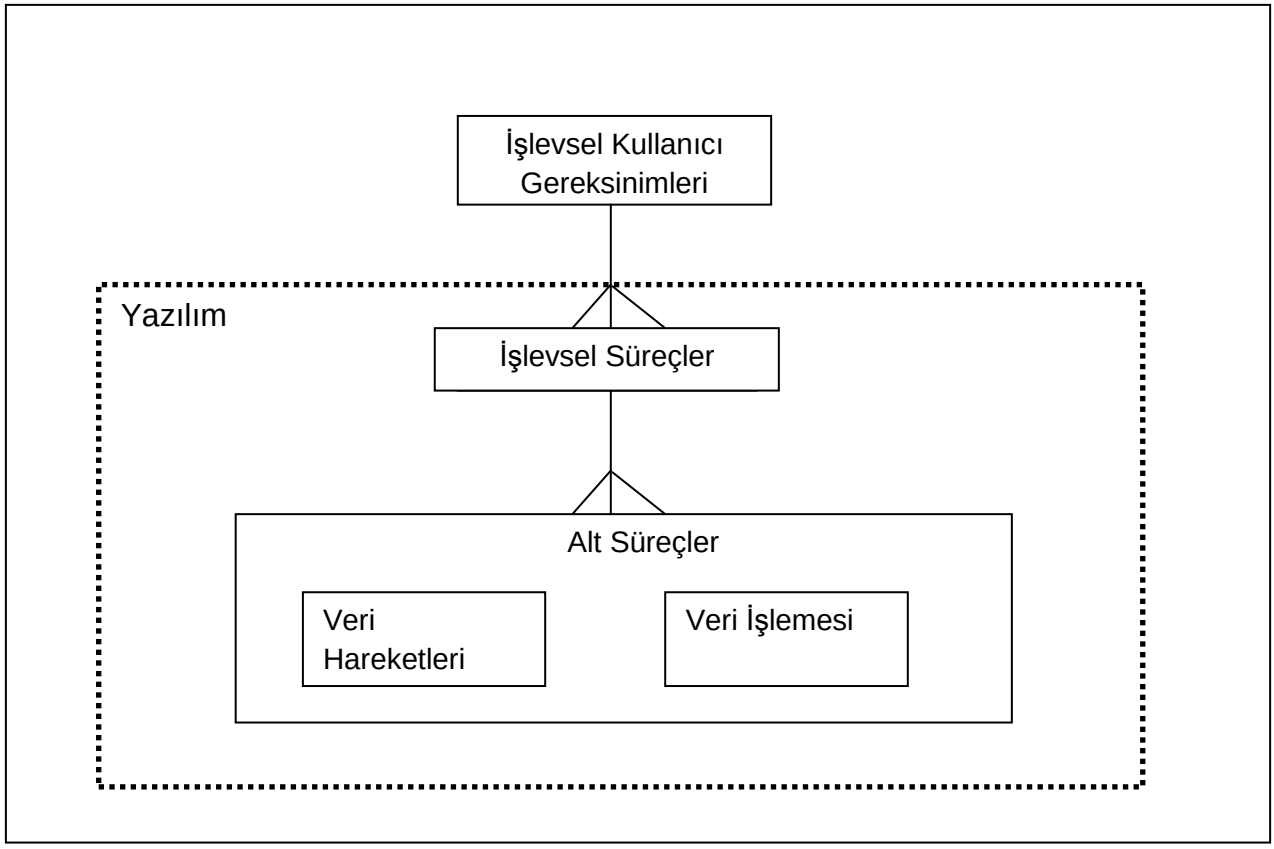


Şekil 3.4 Tetikleyici Olay [7]

Veri grubu, aynı “ilgi odağı” için bütünleştirici bir bakış açısı sağlayan ve iş uygulamalarında genelde özel amaçlı bir sorgu için bir araya gelen verilerdir. Biricik olmalı ve diğer gruplardan ayrımı net bir şekilde yapılabilmelidir. İlgi odağı kavramı COSMIC Ölçme Kılavuzuna göre işlevsel kullanıcı gereksinimleri bakış açısıyla tanımlanan her şeydir. Her bir veri grubu işlevsel kullanıcı gereksinimleri içerisinde bir ilgi odağı ile doğrudan ilişkilidir. Veri tabanı üzerinde işlevsel süreçler aracılığı ile gerçekleşen okuma ve yazma veri hareketleri ile hareket eden veri grubunun, tek bir ilgi odağı ile ilgili veri taşıyıp taşımadığı noktası büyük öneme sahiptir. Gerçekleşen olay sırasında hareket eden veriler birden fazla ilgi odağını ilgilendiriyorsa, ilgilendirdiği ilgi odağı sayısı kadar veri hareketi ölçümde dikkate alınır.

3.3. Ölçme Evresi (“Measurement Phase”)

Bu evrede, tüm işlevsel süreçler için veri hareketleri belirlenir, ölçme fonksiyonu uygulanır, sonuçlar birleştirilir ve sonuçta yazılımın işlevsel büyüklüğü elde edilir.



Şekil 3.5 İşlevsel Süreç ve Alt Süreçler [23]

Bir alt süreç veri hareketi ya da veri manipülasyonudur. Şekil 3.5'te işlevsel gereksinimler, işlevsel süreçler ve alt süreçler arasındaki ilişki gösterilmektedir. Ancak veri manipulasyonları COSMIC kapsamında ayrıca ölçülmez, herhangi bir veri manipülasyonunun ilişkili olduğu veri hareketi üzerinden açıklandığı kabul edilir

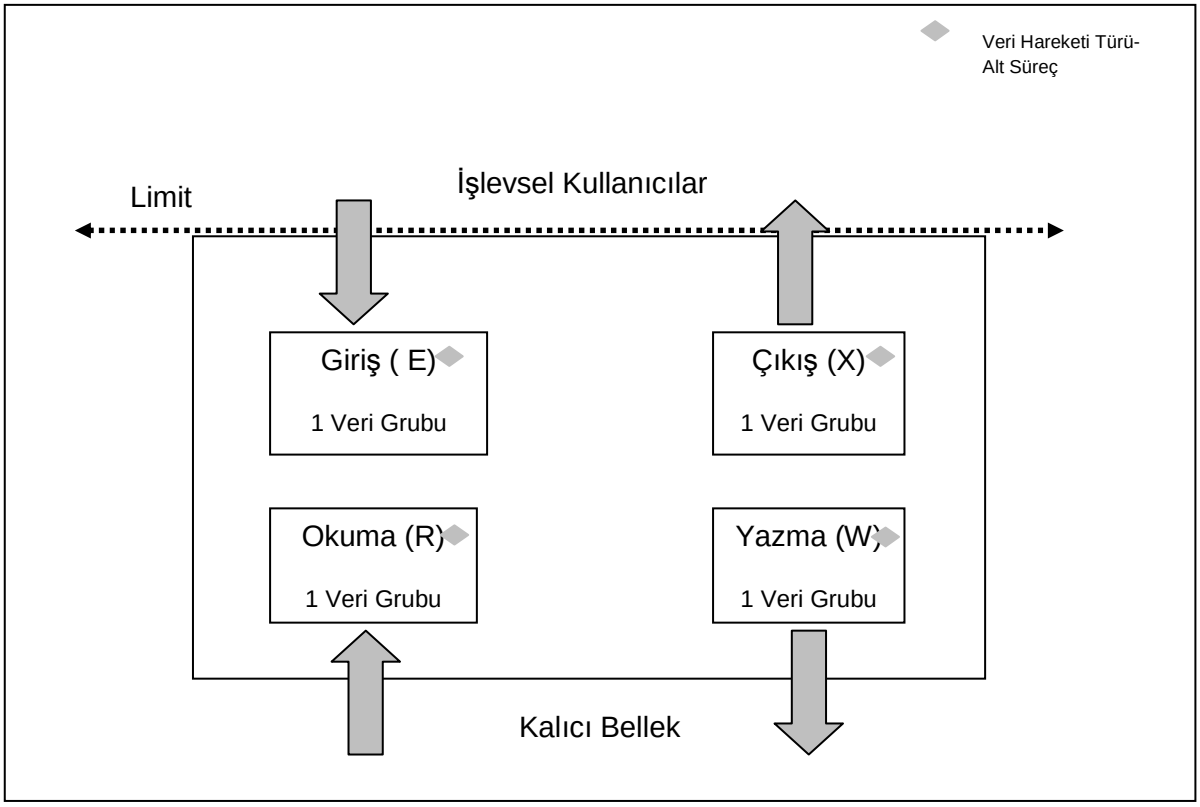
Veri Hareketi; Bir işlevsel sürece ait, yalnızca bir veri grubunun hareket etmesine sebep olan harekettir. COSMIC metodu dört temel veri hareketini ölçme işlemi için temel almaktadır. Söz konusu veri hareketi türleri aşağıda verilmiştir.

- Giriş ("Entry")
- Okuma ("Read")
- Yazma ("Write")
- Çıkış ("Exit")

Veri hareketleri, işlevsel kullanıcılar ve depolama aygıtı arasındaki ilişki Şekil 3.6'da gösterilmiştir:

Giriş veri hareketi, bir veri grubuna ait verilerin, kullanıcı tarafından sınırın diğer tarafına, işlevsel sürecin içerisine yaptığı harekettir. Birden fazla veri grubuna ilişkin veri taşınıyor ise, farklı veri grubu çeşidi kadar hareket tanımlanır. Giriş veri hareketinin ilişkili veri manipülasyonlarını içerdiği düşünülür.

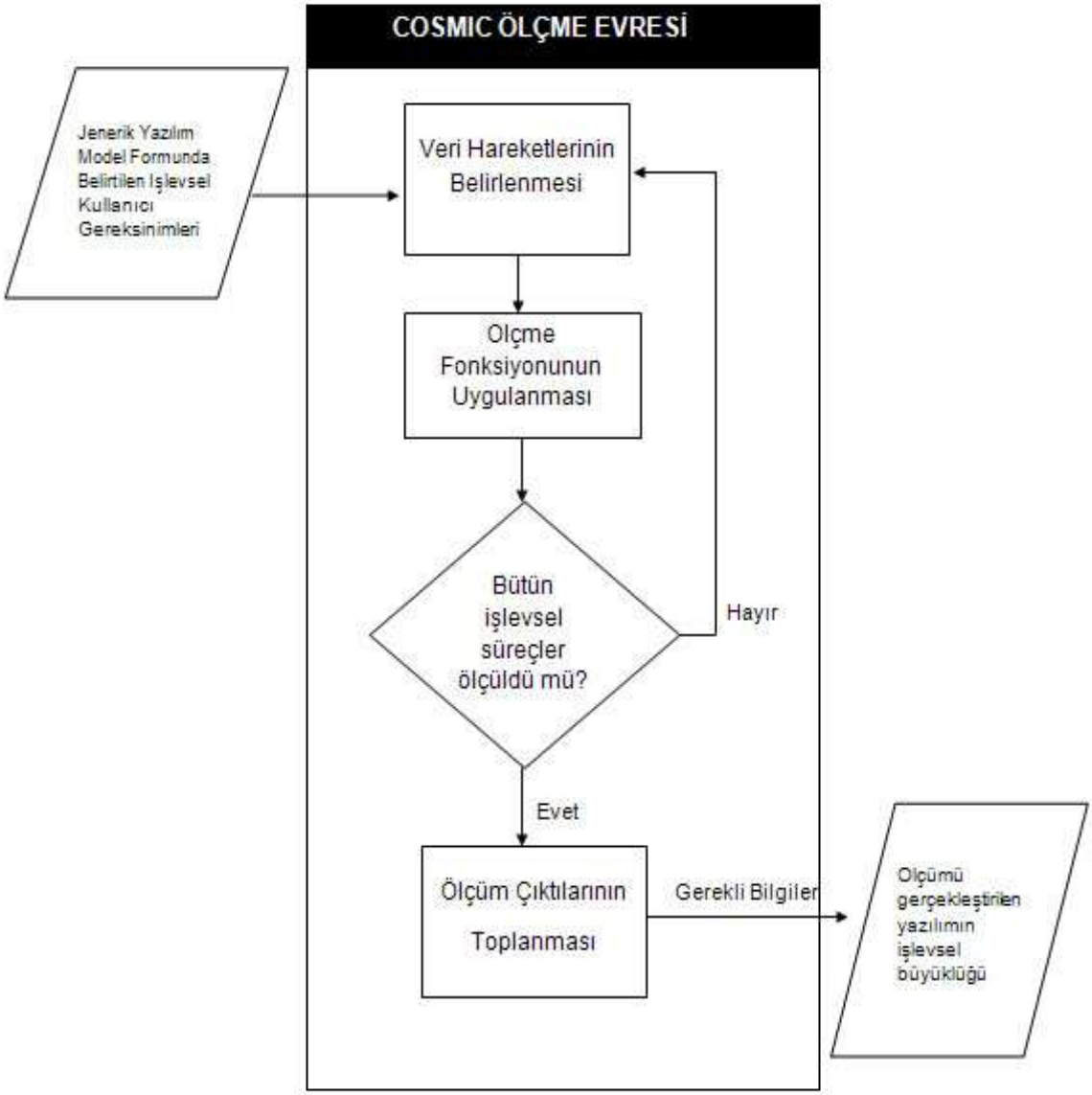
Çıkış veri hareketi, bir veri grubuna ait verilerin, işlevsel süreç içerisinden sınırın diğer tarafına, kullanıcı tarafına yaptığı harekettir. Birden fazla veri grubuna ilişkin veri taşınıyor ise, farklı veri grubu çeşidi kadar hareket tanımlanır. Çıkış veri hareketinin ilişkili veri manipülasyonlarını içerdiği düşünülür.



Şekil 3.6 Veri Hareket Yöntemleri ve Veri Hareket Yöntemlerinin Fonsiyonel Kullanıcılar ve Veri Grupları ile İlişkisi [7]

Okuma veri hareketi, bir veri grubuna ait verilerin, kalıcı bellekten, işlevsel sürecin içerisine yaptığı harekettir. Birden fazla veri grubuna ilişkin veri taşınıyor ise, farklı veri grubu çeşidi kadar hareket tanımlanır. Okuma veri hareketinin ilişkili veri manipülasyonlarını içerdiği düşünülür.

Yazma veri hareketi, bir veri grubuna ait verilerin, işlevsel süreç içerisinden kalıcı belleğe yaptığı harekettir. Birden fazla veri grubuna ilişkin veri taşınıyor ise, farklı veri grubu çeşidi kadar hareket tanımlanır. Yazma veri hareketinin ilişkili veri manipülasyonlarını içerdiği düşünülür.



Şekil 3.7 Ölçme Evresi [7]

Ölçme Fonksiyonunun Uygulanması: COSMIC ölçme fonksiyonu, COSMIC standardının temel aldığı değeri ölçme argümanına atar. COSMIC ölçme fonksiyonunu için söz konusu argüman veri hareketleridir.

COSMIC için standart ölçü birimi olan 1 COSMIC İşlev Puan (CFP) alt süreç seviyesinde bir veri hareketine karşılık gelmektedir. Dolayısı ile bir alt süreç için yapılan büyüklük ölçümünde elde edilen değer, ilgili süreç içerisinde gerçekleşen veri hareketlerinin sayılarının toplamı olarak ifade edilebilir.

$$\text{büyüklük}(\text{işlevsel süreç}_i) =$$
$$\Sigma\,\text{büyüklük}(\text{Giriş}_i) + \Sigma\,\text{büyüklük}(\text{Çıkış}_i) + \Sigma\,\text{büyüklük}(\text{Okuma}_i) + \Sigma\,\text{büyüklük}(\text{Yazma}_i)$$

Birleşik büyüklük hesabı için aynı taneciklik seviyesinde (“level of granularity”) ölçüm yapıldı ise süreç bazında elde edilen değerler matematiksel olarak toplanır, aksi halde aynı taneciklik seviyesine ölçeklendirildikten sonra toplama işlemi gerçekleştirilir. Farklı taneciklik seviyesinde yapılan ölçümlerin ölçeklendirilmesi ile ilgili ayrıntılar “COSMIC Method v3.0 Advanced & Related Topics” [58] belgesinde anlatılmıştır. Ancak bu konu bizim çalışmamızın kapsamı dışında tutulmuştur.

4. İlişkili Çalışmalar

Tez kapsamında yapılan çalışma için incelenen kaynaklar konularına göre birkaç alt başlıkta toplanmıştır;

1) COSMIC metodunu temel alan çalışmalar
2) İşlevsel büyüklük ölçümünün otomatikleştirilmesi ile ilgili çalışmalar
3) Yazılım belirtim dokümanları üzerinden işlevsel büyüklük hesabı yapılan çalışmalar

Yapılan çalışmalar incelendiğinde, konuların birbirleriyle ilişkili olarak ele alınmaları sebebiyle yukarıda belirtilen başlıkların birden fazlasını kapsamı içerisine alan çalışmaların olduğu görülmüştür.

COSMIC metodu kullanılarak yapılan çalışmalar [23][24][25][26], yapılacak tez çalışması için temel oluşturması amacıyla incelenmiştir. [26] kapsamında yapılan çalışmanın aynı zamanda işlevsel büyüklük ölçümünün otomatikleştirilmesi alt başlığı kapsamına girdiği görülmüştür.

Abran [23] yaptığı çalışmada COSMIC ölçme metodunun ortaya çıkışını, çıkış amacını ve temel mantığını anlatmıştır. Ayrıca çalışma içerisinde ölçme prensipleri ayrıntılı olarak açıklanmıştır.

Abran et al. [24] yaptıkları çalışmada COSMIC kullanılarak gerçekleştirilen saha çalışmaları sonucunda ölçülen işlevsel büyüklük ve onunla ilişkili harcanan işgücü değerlerini ele almışlardır. Birbirleriyle ilişkisi olmayan veri örnekleri ile çalışılarak yapılan çalışmalarda, büyüklük ve harcanan işgücü oranı değerleri için elde edilen sonuçlar birbirlerine oldukça yakın çıkmışlardır.

Demirörs et al. [25] çalışmalarında, işlevsel büyüklüğün, iş gücü ve maliyet kestirimi için en kritik bilgi olmasından dolayı, yazılım projelerinde büyük önem taşıdığını ve COSMIC ölçme sonuçlarının ölçümü yapan kişinin bilgi ve dikkatine bağlı olarak değiştiğini belirtmişlerdir. Bu çalışmada çoklu durum çalışması yapılmış ve sıklıkla karşılaşılan hataların neler olduğu ortaya konmuştur. Çalışma sonunda ölçüm standartlarındaki yanlış yorumlamalar ve yöntemlerdeki eksik öğrenmeler, ölçüm sonuçlarının hatalı olmasının en temel nedeni olarak belirlenmiştir.

Lavazza ve Bianco [26] çalışmalarında, UML modellerinin COSMIC tabanlı ölçmeleri desteklediği ve UML dilinin yaygın kullanımı sebebiyle UML tabanlı ölçmelerin

öneminin giderek arttığı belirtilmiştir. Çalışma kapsamında COSMIC kurallarına göre kolayca ölçümü yapılabilen UML modellerinin nasıl oluşturulabileceği üzerinde durulmuş ve bu konuda bir durum çalışması yapılmıştır. Sonuç olarak UML kullanılarak göreli kolay ölçüm yapılabilen modellerin oluşturulabildiği gösterilmiştir.

Yazılım işlevsel büyüklük ölçmenin otomatikleştirilmesi konusunda birçok araştırma [12][13][14][15][16] yapılmış olmasına karşın tam anlamıyla kabul görmüş bir yöntem bulunmamaktadır.

Yazılımın işlevsel büyüklüğünün otomatik olarak ölçülmesi yaklaşımlarından bir tanesi, kod satır sayısını (“line of code - LoC”) temel almıştır. Abran ve Ho [12] çalışmalarında COBOL ile yazılmış bir program için program kesitleme (“program slicing”) adı verilen ters-mühendislik tekniğiyle, kaynak kod üzerinden otomatik olarak büyüklük ölçmeyi hedefleyen bir çatı tanımlamışlardır. Çalışmada model, kaynak kod dosyaları içerisindeki COBOL diline has bazı ayrılmış sözcükler (“reserved words”) yardımıyla veri hareketleri ve veri manipülasyonlarının çıkarımını yaptıktan sonra, belirlenen bu aktiviteler IFPUG standartları [8] ile doğrulanmıştır.

Jones [13] çalışmasında, gereksinim analizi ve tasarım süreci sonucunda elde edilen ve veri sözlüğü içerisinde yer alan bilgi varlıkları üzerinde gerçekleştirilen farklı bir işlev sayma metodu önermiştir. Veri Hacmi Sayma (“Data Volume Counting”) ismi verilen bu yöntemde veri sözlüğü içerisinde yer alacak işlev türleri öncelikle işaretlenmiş, her bir işaretlenmiş öğe için sayma kuralları uygulanmış, çıkan değerler toplanmış ve toplam üzerinde tercihe göre Uygulama Ayarlama Faktörü (“Application Adjustment Factor”) uygulanmıştır. Hacim ölçüm birimi olarak her bir atomik veri öğesi dikkate alınmıştır.

Kusumoto et al. [14] çalışmalarında geleneksel işlevsel büyüklük hesaplamalarına değinmiş ve kaynak kod üzerinden istatistiksel analizlerle otomatik olarak işlev puan hesaplanması ihtimalini değerlendirmişlerdir. Bunun için ekran geçişleri (“screen transition”) üzerine odaklanarak, web uygulamalarının kaynak kodları üzerinden veri (“data”) ve işlembilgi (“transaction”) işlevlerini saymak için bir metot önermişlerdir.

Diab et al. [15] çalışmalarında, birçok şirketin yazılım geliştirme olgunluk modelleri açısından en üst seviyede olabilmek için kaynak ve enerji yatırımı yaptığını, ölçümlerin bu modeller üzerinde giderek önem kazandığını ve bu sebeple Bilgisayar Destekli Yazılım Mühendisliği (“Computer Aided Software Engineering – CASE”) araçlarının, üzerlerinde geliştirilen çok çeşitli belgeler için otomatik ölçme kolaylıkları

sağladıklarını anlatmışlardır. Buradan yola çıkarak çalışmalarında, Rational Rose RealTime (RRRT) modelleri için COSMIC metodu ile yapılan eşlemeler sonucunda otomatik olarak işlevsel yazılım büyüklüğü hesaplayan µcROSE adını verdikleri aracı tanıtmışlardır. Söz konusu çalışmada, COSMIC ve RRRT kavramları arasında Çizelge 4.1 de gösterilen eşleştirmeler kullanılmıştır.

Çizelge 4.1 Kavram Eşleştirmeleri [15] (COSMIC – RRRT)

COSMIC	RRRT
Katman	Kapsül dizisi
Sınır	Kapsül dizisinin kavramsal sınırları
Veri Öbeği	Mesaj veya veri türü
Fonksiyonel Süreç	Bir veya birkaç hareket
Tetikleyici Olay	Dış mesaj alımı
Nitelik	Nitelik
Giriş	Gelen dış mesaj
Çıkış	Giden dış mesaj
Okuma	Hareketin yönlendirilen niteliği
Yazma	Hareketin güncellenen niteliği

Bu aracın temel işlevleri;

a) COSMIC ölçme işlemi için görsel destek sağlamak,

b) XML formatında RRRT modelleri oluşturmak,

c) ölçme işlemi sırasında gereken RRRT varlıklarının çıkarsamasını yapmak,

d) RRRT modeli içerisindeki C++ kodunun analizini yapmak,

e) işlevsel süreçleri, veri gruplarını ve veri hareketlerini belirlemek

f) COSMIC işlevsel büyüklük hesaplamak

g) ölçme sonuçlarını konsolide etmek ve raporlamak

olarak belirtilmiştir.

Çalışmada kural çıkarımı ve doğrulaması yapılırken COSMIC grubundaki uzmanların yardımı alınmıştır. Yapılan durum çalışmaları sonucunda µcROSE ile elde edilen

sonuçlar, uzmanların gerçekleştirdiği ölçüm sonuçları ile karşılaştırılmış ve doğrulanmıştır.

Paton kaynak kod temelli çalışmasında [16], bir programın veri akış tabloları şeklinde ifade edilmesinin işlevsel büyüklük analizinde oldukça kullanışlı olduğunu vurgulamış, bu amaçla programları program kesitlemeye benzer bir yaklaşımla, dinamik ve statik kod analizi yöntemleri kullanarak veri akış tabloları şeklinde ifade ederek işlevsel büyüklüğün değerlendirilebileceğini belirtmiştir.

Diğer bir yaklaşım ise yazılım gereksinim belirtim dokümanlarını (“software specification documents”) analiz ederek işlevsel büyüklük çıkarımı yapmaktır. Bu konuda yapılmış çalışmalarda [59][60][44][61][62][63][64], standart ve geniş kabul görmüş kullanımları dikkate alınarak genel olarak UML belirtimleri kullanılmıştır. Bu çalışmalardan bazılarının [44][61][62] işlevsel büyüklük ölçme işleminin otomatikleştirilmesi kapsamına da girdiği görülmüştür. [64] kapsamında yapılan çalışmanın ise hem işlevsel büyüklük ölçümünün otomatikleştirilmesi hem de COSMIC işlevsel büyüklük ölçüm yöntemi alanında yapılan çalışmalar kapsamına girdiği görülmüştür.

Rask çalışmasında [59], veri-akış (“data-flow”) ve varlık-ilişki (“entity-relationship”) diagramlarının analizine dayanan bir yöntem ile yazılım büyüklüğü ölçmeyi hedeflemiştir. CASE araçlarının ortaya çıkması ve yazılım mühendisliğinin tüm evrelerini kapsayacak şekilde gelişmesiyle ortaya çıkacak verilerin, daha anlamlı ve sonuca yönelik olacağını belirtmiştir.

Gramantieri et al. çalışmalarında [60], işlev puan ölçümü için varlık-ilişki ve veri akış diyagramlarının ortak olarak kullanıldığı bir yöntem önermişlerdir. Bu yöntem varlık-ilişki ve veri-akış diyagramlarını, veri-akış diyagramları içerisindeki veri medyalarını (“stores”), varlık-ilişki diyagramlarındaki varlık ve ilişkiler ile yer değiştirerek entegre etmiştir.

Levesque çalışmasında [44], UML diyagramlarından kullanım durumu ve akış diyagramları kullanılarak COSMIC işlevsel büyüklük hesabının yapılması hedeflenmiştir. Bu çalışmada işlevsel süreç içerisinde iki çeşit hareket ele alınmıştır. Birincisi veri hareketi diğeri ise veri manipülasyonudur. Veri manipülasyonu yazılım karmaşıklığının ölçülmesi ile doğrudan ilişkilidir. UML ve COSMIC eşleme noktasında her bir kullanım durumu bir işlevsel sürece, akış diyagramlarında her bir nitelik bir veri

grubuna ve akış diyagramları üzerindeki mesajlaşmalar veri hareketlerine karşılık gelmektedir.

Bevo et al. çalışmalarında [61], COSMIC işlevsel büyüklük ölçme metodu ile UML (Unified Modeling Language) arasında bir eşleme yapmaya çalışmışlardır. Çalışmada, işlevsel büyüklüğün kullanım durumu diyagramları kullanarak ölçülmesi amaçlanmıştır. Bu eşlemede, her bir "use case"in bir işlevsel sürece, "kullanım durumu" içersindeki karşılıklı etkileşimlerin veri hareketlerine ve "kullanım durumu" senaryolarındaki her bir aktörün bir işlevsel kullanıcıya karşılık geldiği varsayılmıştır. Çalışmanın sonunda yaptıkları eşlemenin sonuçlarını değerlendirmek amacıyla durum çalışmaları gerçekleştirmişlerdir. Metric Xpert isimli aracın kullanıldığı bu durum çalışmalarında, söz konusu araç ve uzmanlar tarafından yapılan ölçümler kıyaslanmış ve değerlendirmeler yapılmıştır.

Jenner'in yaptığı çalışma [62] genel karakteristiği itibarı ile Bevo et al. çalışması [61] ile benzer niteliktedir. Jenner da COSMIC ile UML belirtimleri arasında bir eşleme yapma amacı gütmüştür. Jenner, Bevo et al.'dan farklı olarak işlevsel süreç için kullanım durumu diyagramları yerine akış diyagramlarını temel almıştır. Veri hareketlerini ise akış diyagramlarındaki etkileşim mesajları karşılamaktadır. Jenner da çalışmasını yaptığı durum çalışmalarıyla desteklemiştir.

Poels yaptığı çalışmada [63], COSMIC metodunu, olay tabanlı yaklaşım gibi, kurumsal bilişim sistemleri geliştirme yöntemleri üzerine uygulayabilmek için bazı kurallar tanımlamıştır. Olay tabanlı yaklaşımlarda ve COSMIC yönteminde olay tetikleyicisi kavramı doğal bir eşleme olarak karşımıza çıktığından çalışmada olay ve UML gösterim tabanlı bir metodoloji olan MERODE kullanılmıştır. MERODE iş modeli, işlevsellik modeli ve arayüz modeli olmak üzere farklı model içermektedir. Bu çalışmada söz konusu modellerden "iş modeli" için işlevsel büyüklük ölçümü üzerinde durulmuştur. İş modeli, ilişkili üç alt modelden oluşmaktadır. Bunlar, sınıf diyagramları, nesne-olay tabloları ve durum geçiş tablolarıdır. Çalışmada çıkarımı yapılan eşleme kurallarına göre nesne-olay tablosundaki her bir olay bir işlevsel sürece, her bir nesne türü ise bir veri grubuna karşılık gelmektedir. Bu çalışma için durum çalışmaları yapılmamış olup, çalışma teorik olarak kabul görmüştür.

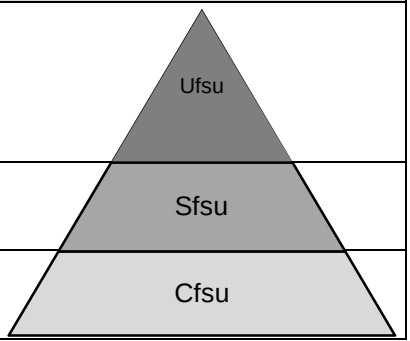
Azzous et al. çalışmalarında [64], RUP ile yapılan geliştirmeler için, işlevsel büyüklüğü otomatik olarak ölçmeye dayanan bir yaklaşım ve araç önermişlerdir. Söz konusu araç COSMIC ile UML kavram ve gösterimleri arasında eşlemeye dayalı

olarak bir değerlendirme yapmaktadır. UML ile COSMIC arasındaki eşlemeye Çizelge 4.2'de yer verilmiştir. Bu çalışmada, tabloda UML karşılığı olmayan "tetikleyici olay"ı kullanım durumu diyagramlarındaki normal mesajlardan ayırabilmek amacı ile yeni bir UML şablonu tanımlanmıştır. Söz konusu otomatikleştirme aracı, geliştirme evresinin üç farklı aşaması için yazılım büyüklüğü hesaplayabilmektedir. İş modelleme ve gereksinim analizi evresinde kullanım durum diyagramları ile analiz ve tasarımın ilk evrelerinde senaryolar üzerinden ve analiz ve tasarımın ileri evrelerinde detaylandırılmış senaryolar üzerinden büyüklük ölçümü sağlayabilmektedir(Çizelge 4.3). Söz konusu çalışmanın doğrulaması Rice Cooker durum çalışması üzerinde yapılmış ve sonuçlar değerlendirilmiştir.

Çizelge 4.2 Kavram Eşleştirmeleri [64] (COSMIC – UML)

COSMIC	**UML karşılığı**
Yazılım Sınırları	"Kullanım Durumu" diyagramları
Yazılım Katmanı	*UML karşılığı bulunmamaktadır*
COSMIC Kullanıcısı	UML aktörü
Fonksiyonel Süreç	(Herbir) "Kullanım Durumu"
Veri Hareketleri	Hareket (mesaj)
Tetikleyici Olay	*UML karşılığı bulunmamaktadır*
Veri Öbeği	UML sınıfı
Veri Niteliği	Sınıf sahipliği

Çizelge 4.3 Geliştirme Evrelerine Göre farklı Kullanılan Büyüklük Birimleri [64]

Geliştirme Adımı	**Kullanılan RUP bileşenleri**	**Birim Kuralları**
• İş Modeli • Gereksinim Analizi	• Kullanım Durumu diagramları	
• Analiz/Tasarım	• Senaryolar	
• Analiz/Tasarım	• Detaylı senaryolar	

5. Otomatik Ölçme Yöntemi

Yazılımların işlevsel büyüklüklerini otomatik olarak ölçmeyi sağlayan “Measurement Kütüphanesi”, COSMIC yöntemini dikkate alarak belirlediğimiz kurallar çerçevesinde gerçekleştirilmiştir. Kütüphane, üç katmanlı mimariye sahip, Java iş uygulamaları içerisinden “import” edilerek ve metotlarının çağrılması için uygulama içine küçük kod eklemeleri yapılarak kullanılabilmektedir. Kütüphanenin kullanımı bütün olarak ele alındığında ise manuel olarak gerçekleştirilen adımlarının da olması sebebiyle, yöntemin yarı-otomatik olarak çalıştığı söylenebilir.

Çalışmada öncelikle, kütüphanenin kullanılması için ölçme amaçları (“purpose”) belirlenmiştir. “Measurement Kütüphanesi”, “Var olan bir yazılımın tamamının ya da belirlenen bazı gereksinimlere karşılık gelen alt birimlerinin işlevsel kullanıcılar için sağlanan işlevselliğinin ölçülmesi” veya “Var olan sistem üzerinde yapılan değişikliklerin işlevsel büyüklüğünün ölçülmesi” amaçları için kullanılabilir.

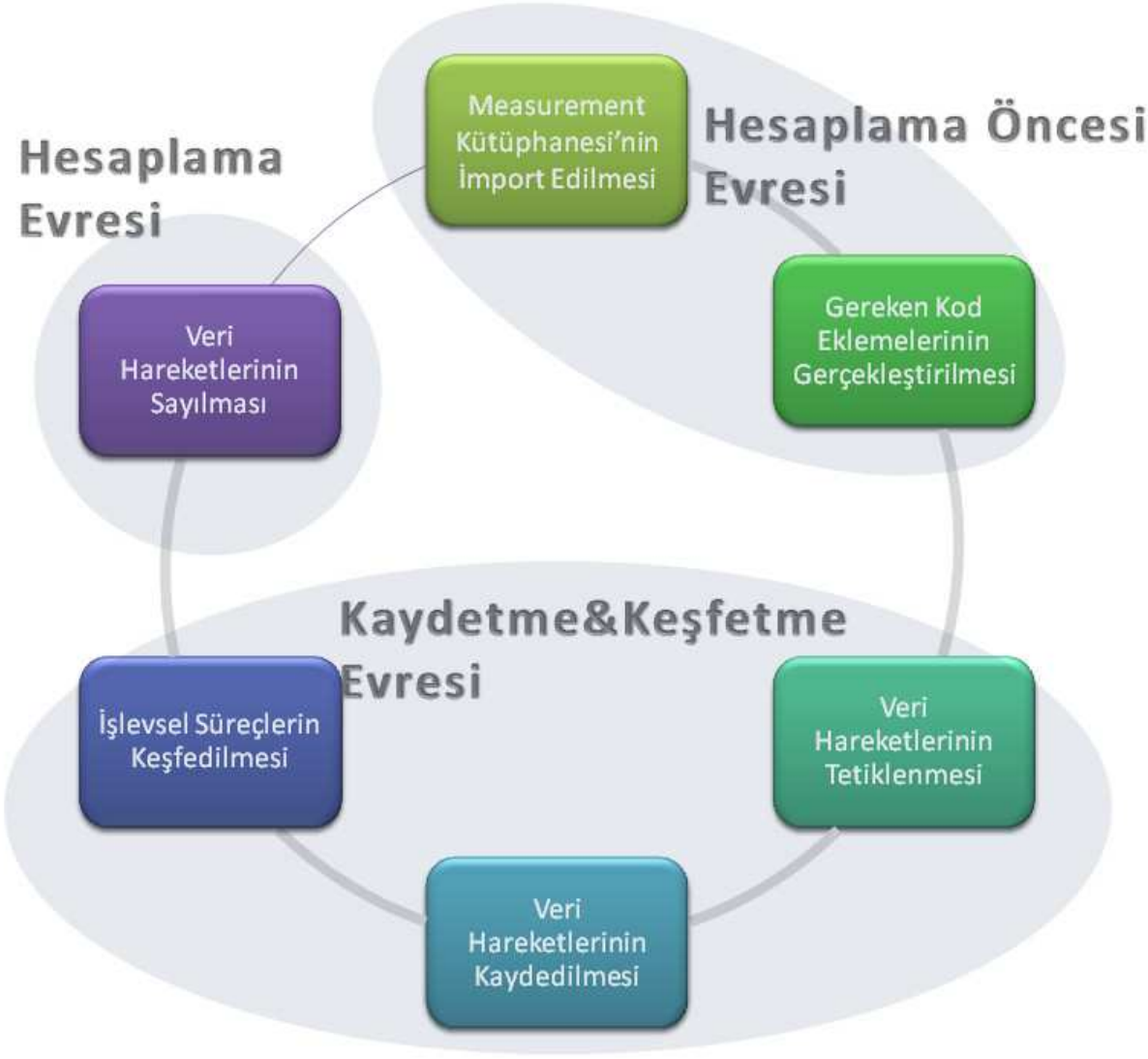
Belirlenen kapsam dahilinde işlevsel kullanıcı; uygulamanın işlevlerini grafik kullanıcı arayüzü (GKA) (“graphical user interface – GUI”) üzerinden tetikleyen, bir kişi ya da diğer bir uygulama olabilir. Ölçmenin kapsamı (“scope”) ise uygulamanın bütününün, belirlenen bir bölümünün ya da bir önceki uygulamaya eklenen kısımlarının işlevsel büyüklüklerinin ölçülmesi olabilir. Ölçme için taneciklik seviyesi olarak, standart taneciklik seviyesi (“standart level of granularity”) olarak kabul edilen işlevsel süreç taneciklik seviyesi (“functional process level of granularity”) benimsenmiştir.

5.1. Ölçme Yöntemi

Measurement Kütüphanesi kullanılarak gerçekleştirilen ölçme eylemi üç ana evrede gerçekleştirilmektedir:

1. Hesaplama Öncesi Evresi
2. Kaydetme ve Keşfetme Evresi
3. Hesaplama Evresi

Bu adımlara ait akış grafiği Şekil 5.1’de verilmiştir:



Şekil 5.1 Ölçme Adımları

Hesaplama Öncesi Evresi: Hesaplama Öncesi evresi Measurement Kütüphanesi kullanılarak gerçekleştirilen ölçme işleminin ilk evresidir. Tamamen manuel olarak gerçekleştirilen bu evre "Measurement Kütüphanesi'nin Import Edilmesi" ve "Gereken Kod Eklemelerinin Gerçekleştirilmesi" olmak üzere iki alt evreden oluşmaktadır.

"Measurement Kütüphanesi'nin Import Edilmesi" evresinde kütüphaneye işlevsel büyüklük ölçümü gerçekleştirilecek olan kod içerisinden referans verilir (import edilir). Bu aşamadan sonra kütüphane ilgili kod için kullanılabilir olacaktır.

"Gereken Kod Eklemelerinin Gerçekleştirilmesi" evresinde Ek 1'de yer alan "Measurement Kütüphanesi Kullanım Kılavuzu" dokümanında belirtilen kod eklemelerinin gerekli yerlere yapılması beklenmektedir. Bu evre yazılımın işlevsel büyüklüğünün doğru olarak ölçülebilmesi açısından en önemli evrelerden biridir.

Kaydetme ve Keşfetme Evresi: Kaydetme ve Keşfetme evresi veri hareketlerinin kaydedilerek işlevsel süreçlerin keşfedildiği evredir. “Veri Hareketlerinin Tetiklenmesi”, “Veri Hareketlerinin Kaydedilmesi” ve “İşlevsel Süreçlerin Keşfedilmesi” olmak üzere üç alt evreden oluşmaktadır.

“Veri Hareketlerinin Tetiklenmesi” evresi, işlevselliğe ait veri hareketlerinin kullanıcı tarafından, uygulama arayüzü üzerindeki (GKA) arayüz bileşenleri vasıtasıyla tetiklenerek bildirildiği evredir. Measurement Kütüphanesi kullanılarak ölçme işleminin gerçekleştirilebilmesi için ölçümü yapılacak tüm işlevselliğin kullanıcı tarafından arayüz üzerinden tetiklenmesi gerekmektedir. Bu evre, sürecin manuel olarak gerçekleştirilen son adımıdır.

“Veri Hareketlerinin Kaydedilmesi” evresi arayüz üzerinden gerçekleştirilen tetiklemeler sonucunda oluşan veri hareketlerinin kaydedildiği evredir. Bu evrede COSMIC’in temel aldığı dört temel veri hareketi (Giriş, Okuma, Yazma, Çıkış), daha sonra işlevsel süreçlerin keşfi için kullanılacak olan “gerçekleştikleri zaman” bilgisiyle birlikte kaydedilmektedir. Bu evre aslında “Veri Hareketlerinin Tetiklenmesi” evresiyle iç içedir ve bu evre ile zaman paylaşımlı olarak gerçekleşmektedir.

“İşlevsel Süreçlerin Keşfedilmesi” evresi önceki evrede kaydedilen zaman bilgilerinin ve veri hareketlerinin analiz edilerek, ilgili hareketlerin işlevsel süreç yaratıp yaratmadığının otomatik olarak belirlendiği evredir. COSMIC için bir işlevsel süreç, en azından bir Giriş veri hareketi ve bunu izleyen bir Yazma ya da Çıkış hareketinden oluşmaktadır [7]. Dolayısıyla burada öncelikli şart, bir Giriş veri hareketinin gerçekleşmiş olmasıdır. Bu şart sağlandığında bu hareketi takip eden diğer hareketler incelenir. Ard arda gerçekleşen bir hareket zincirinin işlevsel süreç niteliği taşıması için, veri hareketlerinin yukarıdaki kuralı sağlayacak şekilde (en azından GY ya da GÇ) birbirini izlemesi ve belirlenen süre içerisinde gerçekleşmesi gerekmektedir. Measurement Kütüphanesi kapsamında bu süre 1 saniye olarak belirlenmiştir. Bu değerlendirme sonucunda işlevsel süreç olarak belirlenen hareket zinciri kayıt altına alınır. Burada önemli nokta, işlevsel sürecin tetikleyicisinin de bu sırada kaydedilerek daha sonra aynı işlevsel sürecin tekrar tetiklenmesi sonucu oluşabilecek mükerrer kaydın önüne geçilmiş olunmasıdır.

Hesaplama Evresi: Veri hareketlerinin sayılıp ardından COSMIC ölçme fonksiyonun uygulandığı evredir. Önceki evrede işlevsel süreç keşfi ile belirlenen süreçler

içerisindeki hareketler otomatik olarak sayılır ve sonrasında COSMIC Measurement Manual'da [7] belirtildiği şekilde her bir veri hareketi 1 CFP olarak hesaplanır. Tüm işlevsel süreçler için işlev puan değeri tek tek hesaplandıktan sonra, hesaplanan değerler toplanarak uygulamanın bütünü için işlev puan değeri, CFP cinsinden elde edilir.

5.2. Ölçme Kütüphanesi

"Measurement" kütüphanesinin UML tanımı Şekil 5.2'de verilmiştir. Kütüphanenin doğrudan ölçme ile ilgili metotları aşağıda açıklanmıştır. Bu metotların uygulama üzerinde nasıl kullanılacaklarına ilişkin ayrıntılı bilgi sonraki bölümde verilmiştir.

Şekil 5.2 "Measurement" kütüphanesinin UML ile temsili

NotifyEntryMovement(): GKU üzerindeki bileşenler için tanımlı hareket dinleyiciler tarafından Giriş ("Entry") hareketlerinin algılanması için çağrılır. İlgili olayı parametre olarak alır ve buradan yola çıkarak tetikleyici kimliğine ulaşır. Tetikleyici kimlikleri

saklanarak ölçme sırasında aynı kaynaktan gelen birden fazla hareket için mükerrer kayıt yapılmasının önüne geçilir. Yine bu metot içerisinde, işlevsel süreç keşfi için kullanılacak tetiklenme zamanı ve hareket türü (Giriş) de kaydedilmektedir.

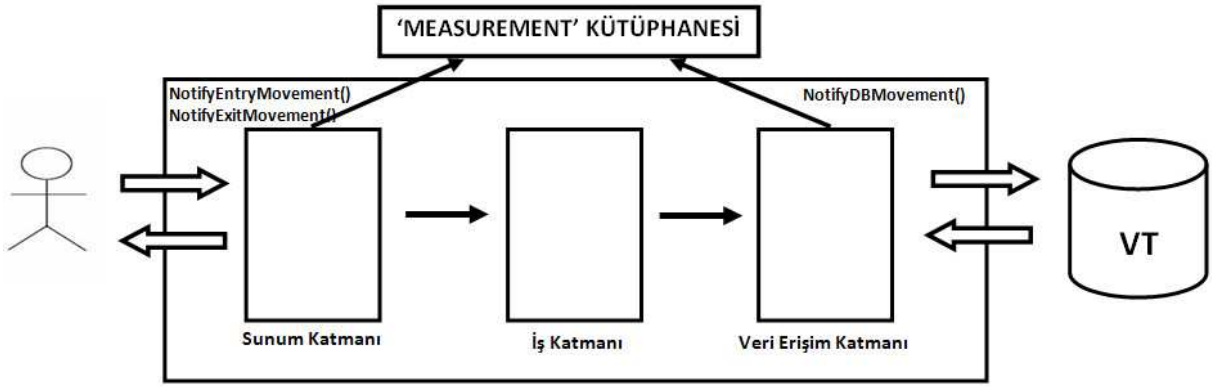
NotifyDBMovement(): GKU üzerinden gelen hareketlerin veri tabanına erişim yaptığı noktalarda sorgu cümleciği parametresi ile çağrılır. Öncelikle gerçekleşme zamanı kaydedilir ve bu zaman, son gelen Giriş hareketinin gerçekleşme zamanı ile karşılaştırılır. Geçen süre belirlenen sürenin (1 sn) altında ise işlevsel süreç oluşturabilecekleri varsayılır. Sonrasında sorgu cümlesi parçalanarak, sorgunun Okuma (“Read”) ve Yazma (“Write”) hareketlerinden hangisini gerçekleştirdiği anlaşılır ve kayıt altına alınır.

NotifyExitMovement(): Veri tabanından dönen değerlerin GKU üzerinde gerçekleştirdikleri hareketler, değişim dinleyicileri yardımıyla dinlenir. Herhangi bir değişim gerçekleştiğinde bu metot çağrılır. Gerçekleşme zamanına göre bir işlevsel sürece ait olup olmadığı belirlenir ve Çıkış (“Exit”) hareketi kayıt işlemi gerçekleştirilir.

CalculateCosmic(): GKU’dan çıkış sırasında çağrılır. Uygulama çalışırken kayıt altına alınan ve işlevsel süreç oluşturduğu doğrulanan hareketler kayıtlardan okunarak sayılır. COSMIC yaklaşımında belirlendiği şekilde, her bir hareket 1 CFP olarak hesaplanır. Sonuçlar kullanıcıya; GKU’nun sunduğu işlevsellik içerisinde keşfedilen işlevsel süreçler, süreçlerin hangi uygulama ekranları üzerinde gerçekleştikleri, bu süreçlerin içerisinde yer alan veri hareketi türleri ve ölçülen yazılım büyüklüğü olarak rapor halinde sunulur.

Bu çalışmada önerilen yöntemle doğru bir hesaplama sağlamak için, işlevsel büyüklük ölçümü yapılmak istenen tüm işlevlerin, GKU üzerinde kullanıcı tarafından en az bir kere manuel olarak işletilmiş olması gerekir. Ölçme süresince hiç aktif olmayan işlevlerin büyüklüğü, “Measurement Kütüphanesi”nin yeteneği dahilinde olsa da kullanıcı tarafından bildirilmediğinde kütüphane tarafından hesaplanamaz.

Ölçme işleminin uygulanacağı üç katmanlı mimari ve bu mimarinin, ölçme işlemi için entegre edilen Measurement Kütüphanesi ile ilişkisi, Şekil 5.3’te verilmiştir.



Şekil 5.3 Otomatik Ölçme Yönteminin Mimarisi

5.3. Otomatik Ölçmenin Kısıtları

Yukarıda detayları tanımlanan, uygulamanın çalışması sırasında COSMIC işlevsel büyüklüğü otomatik olarak ölçme yönteminin, gerçekleştirimi ve kullanımıyla ilgili bazı kısıtlara değinmek gerekir.

İlk olarak, bu yöntemi izleyerek kütüphanenin kullanımı ile sadece GKU üzerinden tetiklenen işlevselliğin ölçümü otomatik olarak yapılabilir. Uygulamanın iletişimde olduğu diğer uygulamalar varsa ve bunlar tarafından tetiklenen işlevler GKU üzerinden çağrılmıyorsa bu işlevlerin büyüklüğü yöntemle hesaplanamaz.

İkinci olarak, kütüphane kapsamında Java Swing sınıfına ait bileşenlerden bazıları dikkate alınmıştır. Bu bileşenler dışındaki bileşenler üzerinden tetiklenen eylemler, kütüphane kullanılarak yapılan otomatik ölçme işlemi sırasında dikkate alınmayacaktır. Kütüphanenin ölçme sırasında dikkate aldığı bileşenler şunlardır:

- JButton
- JTextField
- JComboBox
- JMenuItem
- JList
- JDialog
- JFrame
- JTabbedPane
- JTable

Üçüncü olarak, büyüklüğü hesaplanacak işlevselliğin uygulamanın çalışması sırasında GKU üzerinde en az bir kez işletilmesi gerekmektedir. GKU üzerinde bir kez işletilmeyen işlevlerin büyüklüğü, yöntemle hesaplanamaz.

Dördüncü olarak, GKU üzerinden tetiklenen eylemlerin başlattığı hareketlerin bir işlevsel süreç oluşturup oluşturmadığı, diğer bir deyişle işlevsel süreç keşfi, hareketlerin zaman bilgisini kullanarak yapılmaktadır. Kütüphanenin keşif operasyonunda bu zaman aralığı 1 saniye olarak belirlenmiştir. Ne var ki üç mantıksal katmanlı uygulamanın fiziksel mimarisi, veri hareketlerinin gerçekleşme zamanı üzerinde etkili olabilir. Bu durumda işlevsel süreçler yanlış keşfedilebilir. Bu kısıta çözüm olarak, işlevsel süreç keşfi için zaman aralığı parametresinin dışarıdan verilmesi ya da işlevsel süreç keşfinde zaman bilgisinin kullanımı dışında bir yaklaşımın önerilmesi düşünülebilir. Zaman aralığı parametresi t'nin dışarıdan verilmesi durumunda, GKU üzerinden tetiklenen iki işlevin tetiklenme zamanları arasındaki sürenin t'den büyük olması, işlevsel büyüklüğün doğru hesaplanması için yeterli olacaktır.

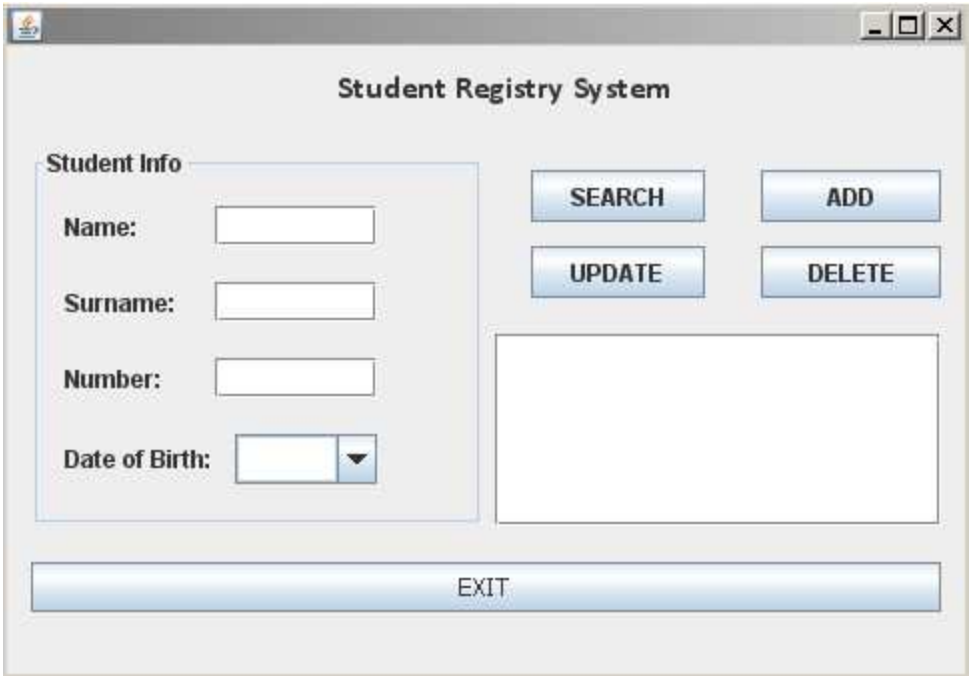
Beşinci olarak, COSMIC yönteminde birden fazla ilgi odağını ("object of interest") ilgilendiren veri hareketlerinin, ilgilendirdiği ilgi odağı sayısı kadar veri hareketi oluşturduğu kabul edilmektedir. Ancak "Measurement Kütüphanesi"nin kullanımını sağlayan ve manuel olarak yapılması önerilen kod eklemelerinde (Ek-1) bu nokta dikkate alınmamaktadır. Yapılan uygulamalar bir işlevsel süreçte tek bir ilgi odağını ilgilendiren verilerin taşındığı uygulamalardır. Karmaşık uygulamalarda kütüphanenin kullanımı için bu husus dikkate alınarak yöntem iyileştirilebilir. Bu tezde yapılan uygulama çalışmalarında odak, yöntemin işlevsel büyüklüğün hesaplanmasında işgücü tasarrufu sağlayıp sağlamayacağını değerlendirmek olduğundan, bu iyileştirme gelecek çalışmalara bırakılmıştır.

6. UYGULAMALAR

Bu bölümde Measurement Kütüphanesi'nin uyguanabilirliğini ve maliyet-etkinliğini değerlendirmek için yapılan çalışmalara yer verilmiştir. Öncelikle yapılan "ön uygulama" ile Kütüphane'nin uygulanabilirliği ele alınmış, elde edilen başarılı sonuçlar ile birlikte kütüphane geliştirilerek daha büyük uygulamalar üzerinde "durum çalışmaları" yapılmış ve elde edilen sonuçlar tartışılmıştır.

6.1. Ön Uygulama

Ön uygulama için kullandığımız uygulama; temel bazı işlevleri gerçekleştiren, Java ile kodlanmış, üç katmanlı bir Öğrenci Veri Tabanı (OVT) uygulamasıdır. Uygulama kapsamında gerekli bilgiler girilerek öğrenci kaydedilebilmekte, silinebilmekte, aranabilmekte ve öğrenci bilgileri güncellenebilmektedir (Şekil 5.1). İşlemlerle ilgili hata ve başarı mesajları GKU üzerinde görüntülenmektedir.



Şekil 6.1 Örnek Uygulamanın Ekran Görüntüsü

Hesaplama öncesinde, "Measurement" kütüphanesi OVT'nin GKU ve VT ile ilişkili sınıfları içerisinden "import" edilmiş ve bu sınıfların içine sırasıyla, aşağıda tanımlanan eklemeler yapılmıştır.

Öncelikle, GKU üzerinde manuel olarak gerçekleşen hareketler ile tetiklenen ActionPerformed() metodu içerisine "Measurement.NotifyEntryMovement()" çağrısı

eklenmiştir. Burada eklenen kod satır sayısı, GKU'nun tüm eylemleri tek bir metot altında ele alındığı için 1'dir.

İkinci adım olarak, veri tabanı sorgusunun işletildiği tüm ExecuteQuery() ve ExecuteQueryUpdate() metotları altına "Measurement.NotifyDBMovement()" çağrısı eklenmiştir. Burada eklenen kod satır sayısı ise ölçme işleminin uygulandığı yazılımın veri tabanı üzerinde çalıştırdığı sorgu adediyle aynıdır. OVT uygulamasında gerçekleşen hareketler dikkate alındığında 4 farklı erişim yapıldığı görülmektedir; dolayısı ile burada veritabanı katmanına eklenen kod satır sayısı 4 olmuştur.

Üçüncü adım olarak GKU üzerindeki bileşenler üzerinde bir değişme olduğu zaman tetiklenen propertyChange() metodu altında "Measurement.NotifyExitMovement() metodu çağrılmıştır. Bu durumda eklenen kod satır sayısı 1'dir. Ancak bunun için, PropertyChangeListener arayüzünün GKU tarafından gerçekleştirilmesi ("implement" edilmesi) gerekmektedir. Ön uygulamada bu gerçekleştirmenin maliyeti 5 satır olmuştur.

Dördüncü ve son olarak, GKU'daki "Kapat" butonu hareketinin de algılandığı ActionPerformed() metodu içerisinde "Measurement.CalculateCosmic()" metodu çağrılmıştır. Bunun için eklenen kod satır sayısı ise 2'dir.

Ölçme işlemine hazırlık aşamasına genel olarak bakıldığında, ölçümü yapılacak uygulamaya eklenecek kod sayısının kabul edilir düzeyde olduğu görülmektedir. Ön uygulamanın ölçümü için eklediğimiz kod satır sayısı 12'dir. Uygulamanın toplam kod satır sayısının 586 olduğu göz önüne alındığında, eklenen kod sayısının toplam koda oranının (%2) oldukça düşük olduğu görülmektedir. Veri tabanı erişimi dışındaki noktalar için, kod ekleme maliyetinin artan kod sayısından bağımsız olduğu düşünüldüğünde, kod satır sayısı çok daha fazla olan uygulamalar için bu oranın %1'ler seviyesine ineceği öngörülmektedir.

Uygulama sonucunda elde edilen değerler incelendiğinde manuel yapılan ölçme ile otomatik olarak yapılan ölçme değerlerinin birbirlerine oldukça yakın olduğu gözlemlenmiştir. Elde edilen sonuçlar Manuel (Çizelge 5.1) ve Otomatik (Çizelge 5.2) olarak ayrı tablolarda gösterilmiştir.

Otomatik ve manuel olarak yapılan ölçümler sonucunda elde edilen değerler karşılaştırıldığında 1/13 oranında bir hata payı olduğu görülmektedir. Örnek uygulamanın kapsamının dar tutulması ve zaman kontrolü ile gerçekleştirilen işlevsel

süreç keşfi, bu denli doğru sonuçlara ulaşmamızı sağlamıştır. Çıkış veri hareketi değerindeki sapma, GKA üzerinde gerçekleşen değişiklikleri izleme noktasında bazı eksiklerimizin olduğunu göstermektedir. Buna rağmen, bu çalışmadan elde ettiğimiz ilk sonuçlar, sonraki çalışmalar için motive edici olmuştur.

Çizelge 6.1 Manuel Ölçme Sonuçları

Manuel Ölçme Sonuçları		**Süreç Türü**			
		Ekle	**Sil**	**Güncelle**	**Ara**
Veri Hareketi Türü	Giriş	1	1	1	1
	Çıkış	1	1	1	2
	Okuma	-	-	-	1
	Yazma	1	1	1	-

Çizelge 6.2 Otomatik Ölçme Sonuçları

Otomatik Ölçüm Sonuçları		**Süreç Türü**			
		Ekle	**Sil**	**Güncelle**	**Ara**
Veri Hareketi Türü	Giriş	1	1	1	1
	Çıkış	1	1	1	1
	Okuma	-	-	-	1
	Yazma	1	1	1	-

6.2. Durum Çalışmaları

Measurement Kütüphanesini uyguladığımız ve kullanılabilirliğini bir ölçüde kanıtladığımız Öğrenci Veri Tabanı, bizi “Measurement” Kütüphanesi’nin daha büyük uygulamalar için kullanılabilirliğini belirlemek ve iş gücü ve zaman kazanımını değerlendirmek konularında cesaretlendirdi.

Measurement Kütüphanesi’nin yazılım projeleri üzerinde uygulanması iki farklı biçimde olabilmektedir.

1- Measurement Kütüphanesi yazılım geliştirme sürecine entegre edilerek kullanılabilir. Böylelikle yazılımın cari durumu sürecin her adımında değerlendirilerek projenin ilerleyen aşamaları için maliyet analizleri yapılabilir, projenin plana uygunluğu değerlendirilebilir ve bu hususta gereken aksiyonlar alınabilir.
2- Measurement Kütüphanesi yazılım geliştirme süreci tamamlandıktan sonra yazılıma entegre edilebilir. Bu şekilde yazılımın nihai durumu ile hedeflenen son ürün değerlendirilerek ihtiyaçların ne kadarının karşılandığı değerlendirilebilir, sonraki çalışmalar için temel alınacak maliyet çıkarımları yapılabilir.

Durum çalışmaları yukarıda sözü geçen uygulama biçimlerini kapsayacak şekilde, aşağıdaki araştırma sorularını yanıtlamak üzere gerçekleştirilmiştir.

AS-1: Önerilen yöntem yeni geliştirilen uygulamaların işlevsel büyüklük ölçmesinde maliyet-etkin midir?

AS-2: Önerilen yöntem daha önceden geliştirilen uygulamaların işlevsel büyüklük ölçmesinde maliyet-etkin midir?

6.2.1. Durum Çalışmaları Tasarımı

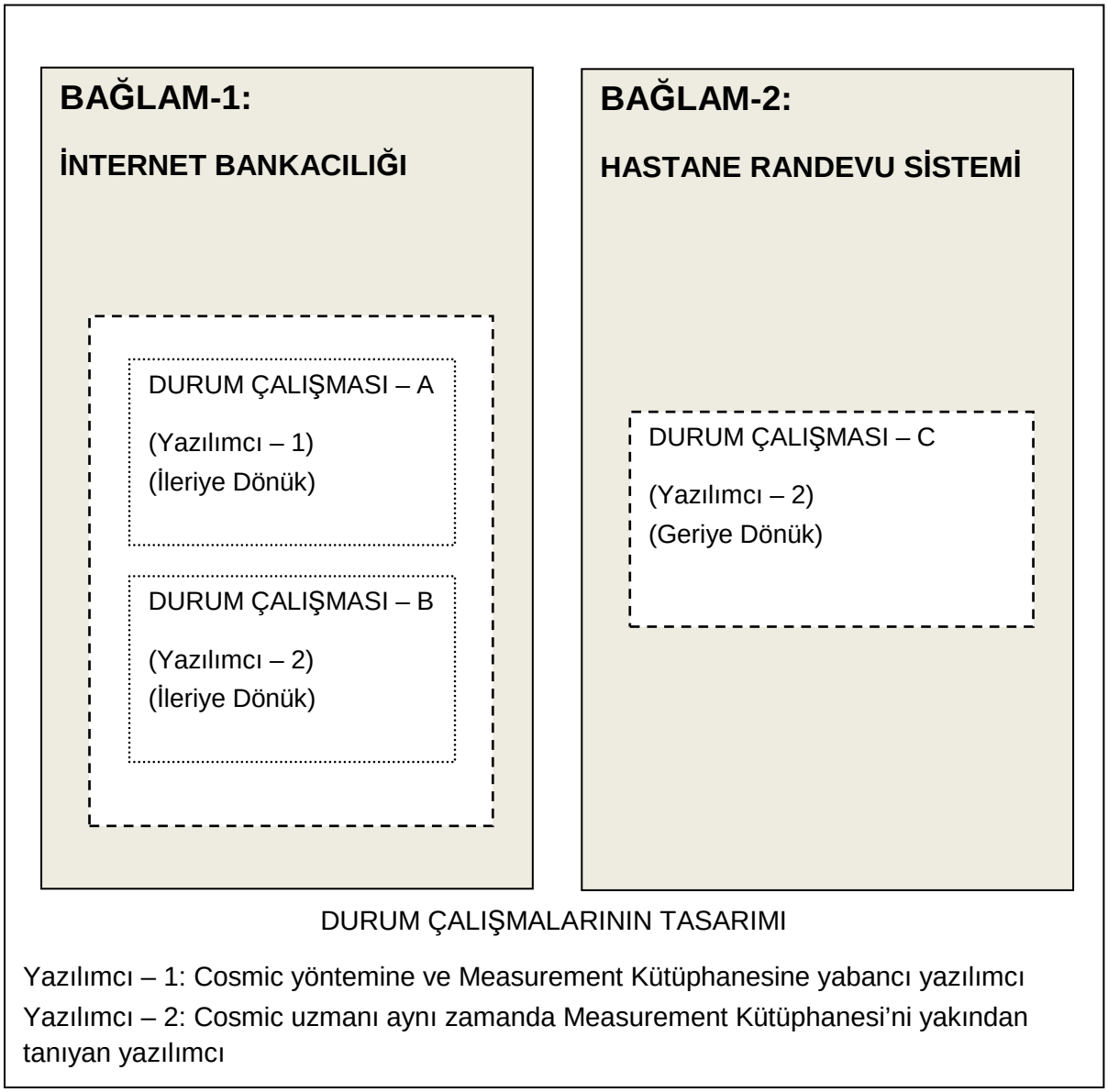
Yuklarıdaki araştırma soruları, durum çalışmalarının yapılabilirliği gözetildikten sonra, aşağıdaki şekilde özelleşmiştir.

AS-1.1: Önerilen yöntem COSMIC yöntemine yabancı biri tarafından kullanıldığında, yeni geliştirilen bir uygulamanın işlevsel büyüklük ölçmesinde maliyet-etkin midir?

AS-1.2: Önerilen yöntem COSMIC yöntemine aşina biri tarafından kullanıldığında, yeni geliştirilen bir uygulamanın işlevsel büyüklük ölçmesinde maliyet-etkin midir?

AS 2.1: Önerilen yöntem uygulamaya yabancı biri tarafından kullanıldığında, daha önceden geliştirilen uygulamanın işlevsel büyüklük ölçmesinde maliyet-etkin midir?

Bu soruları yanıtlayabilmek için, Şekil-6.2'de gösterildiği gibi iki farklı bağlamda, üç farklı durum çalışması tasarlanmıştır. Bunlardan A ve B ileriye dönük ("prospective") çalışmalar, C ise geriye dönük ("retrospective") çalışmadır.



DURUM ÇALIŞMALARININ TASARIMI

Yazılımcı – 1: Cosmic yöntemine ve Measurement Kütüphanesine yabancı yazılımcı
Yazılımcı – 2: Cosmic uzmanı aynı zamanda Measurement Kütüphanesi'ni yakından tanıyan yazılımcı

Şekil 6.2 Durum Çalışmaları Tasarımı

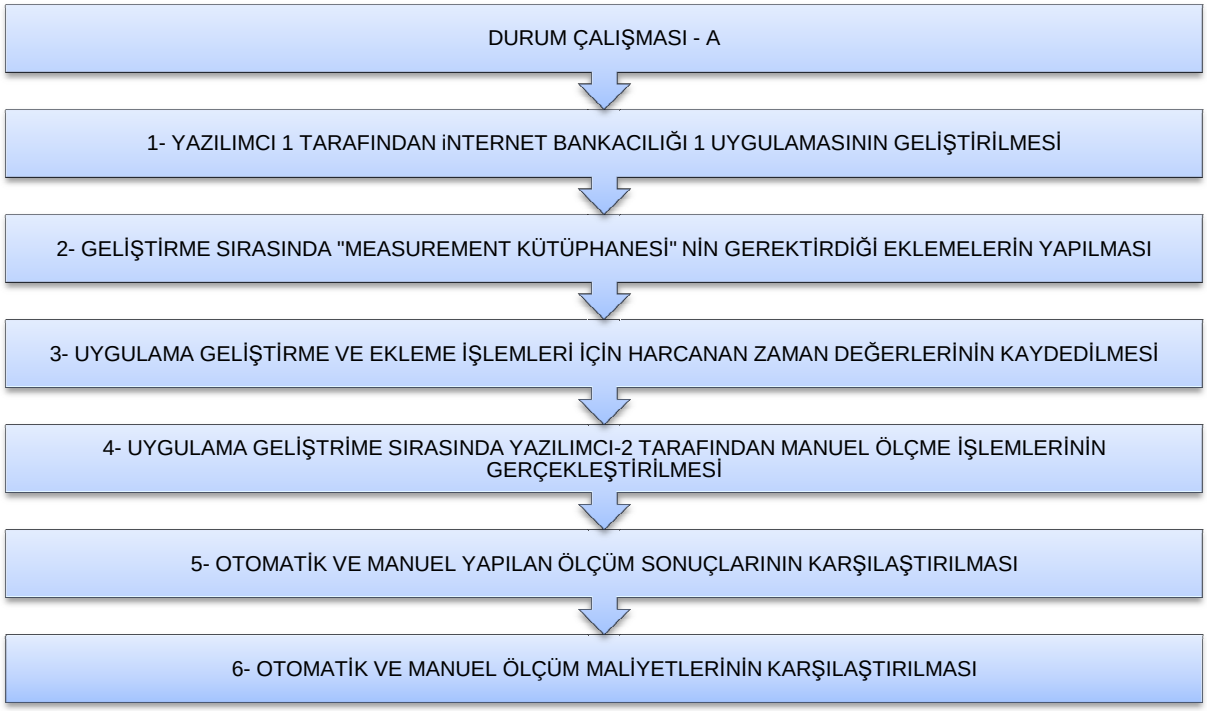
İlk durumda (durum çalışması A), işlevsel büyüklük ölçme konusunda bilgisi olmayan bir yazılımcı (Yazılımcı-1) Measurement Kütüphanesi'ni, gerçekleştirimini yaptığı "İnternet Bankacılığı-1" uygulaması içinden geliştirme surecinde kullanmıştır. Bu durum çalışmasının gerçekleştirilmesindeki amaç COSMIC İşlevsel Büyüklük ölçme metodu ve Measurement Kütüphanesi hakkında bilgisi olmayan bir yazılımcının, Kütüphane'yi kullanarak manuel yapılan ölçmeye göre maliyet-etkin bir şekilde işlevsel büyüklük ölçümü gerçekleştirebildiğini göstermektir. Gerçekleştirilen durum çalışması sırasında, Kütüphane'nin kullanılmaması durumunda katlanılacak manuel ölçme maliyetlerinin de değerlendirilebilmesi amacıyla, projenin farklı evrelerinde Yazılımcı-2 tarafından manuel ölçümler gerçekleştirilmiş olup harcanan süreler kaydedilmiştir.

İkinci durumda (durum çalışması B) işlevsel büyüklük ölçme konusunda bilgisi olan özellikle COSMIC ölçme yöntemi konusunda tecrübeli ve Measurement Kütüphanesi'nin kullanımına hakim bir yazılımcı (Yazılımcı-2) Measurement Kütüphanesi'ni gerçekleştirimini yaptığı "İnternet Bankacılığı-2" uygulaması içinden, yine geliştirme sürecinde kullanmıştır. Geliştirilen bu uygulama "Durum Çalışması A" kapsamında gerçekleştirilen uygulamanın kullanıcıya sağladığı işlevsellik dikkate alınarak seçilmiş, karşılaştırmaların daha sağlıklı yapılabilmesi için uygulamanın aynı işlevsellikte olacak şekilde geliştirilmesi sağlanmıştır. Bu durum çalışmasının gerçekleştirilmesindeki amaç COSMIC İşlevsel Büyüklük ölçme metodu ve Measurement Kütüphanesi hakkında bilgisi olan bir yazılımcının, kütüphaneyi kullanarak manuel yapılan ölçmeye göre maliyet-etkin bir şekilde, işlevsel büyüklük ölçümü gerçekleştirebildiğini göstermektir. Gerçekleştirilen durum çalışması sırasında, kütüphanenin kullanılmaması durumunda katlanılacak manuel ölçme maliyetlerinin de değerlendirilebilmesi amacıyla, projenin farklı evrelerinde Yazılımcı-2 tarafından manuel ölçümler gerçekleştirilmiş olup harcanan süreler kaydedilmiştir. Aynı zamanda bu iki durum çalışması neticesinde elde edilen sonuçlar değerlendirilerek COSMIC ve Measurement Kütüphanesi konularında bilgi sahibi olmamanın kütüphanenin kullanım başarımını ne ölçüde etkilediği analiz edilmiştir.

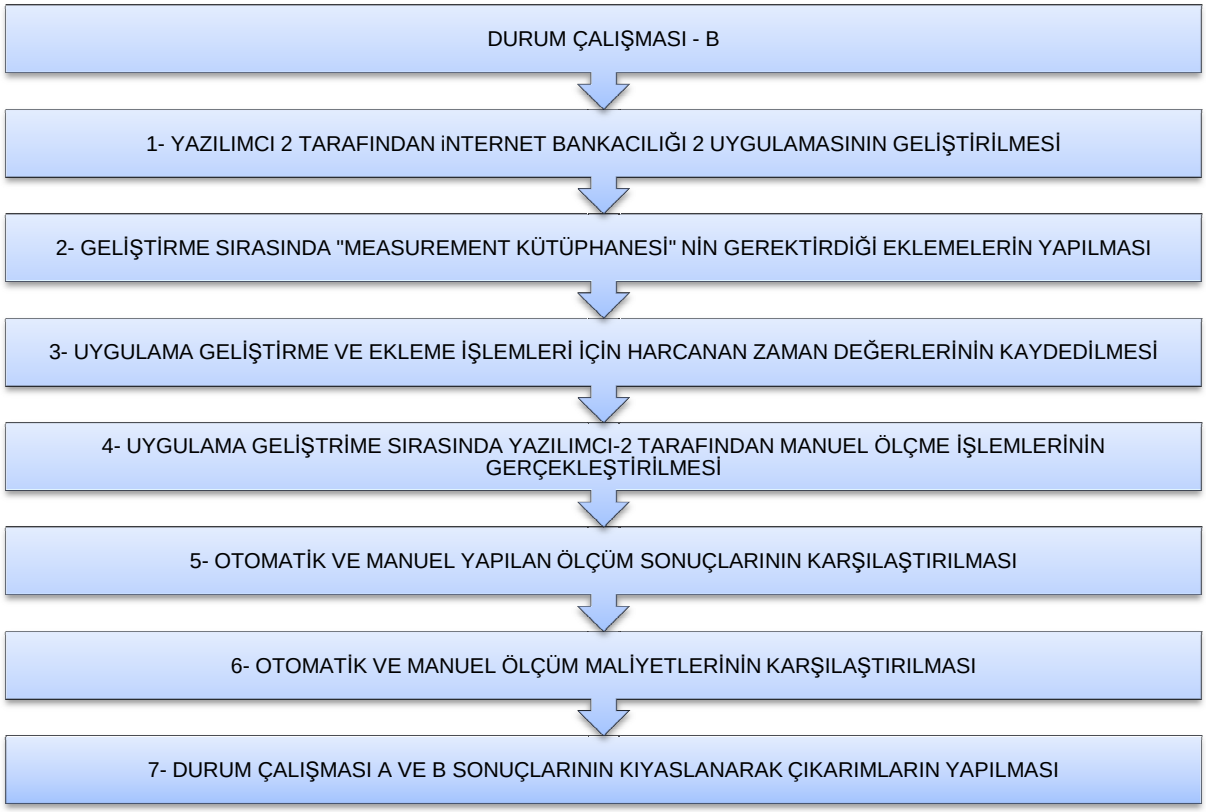
Üçüncü durumda (durum çalışması C), Hacettepe Üniversitesi BİL342 dersi projesinde geliştirilmiş "Hastane Randevu Sistemi" uygulaması üzerine, Yazılımcı-2 tarafından Measurement Kütüphanesi'nin kullanımını sağlayan kod parçaları geliştirme tamamlandıktan sonra manuel olarak entegre edilmiştir. Bu durum çalışmasının gerçekleştirilmesindeki amaç, Measurement Kütüphanesi'nin uygulama geliştirme süreci tamamlandıktan sonra uygulamaya entegre edilerek, manuel yapılan ölçüme göre maliyet-etkin bir şekilde işlevsel büyüklük ölçümü yapıldığının gösterilebilmesidir. Bu uygulama için de diğer durum çalışmalarında yapıldığı gibi manuel ölçüm gerçekleştirilmiş olup sonuçlar kaydedilmiştir. Böylelikle manuel ve otomatik yapılan işlevsel büyüklük ölçümleri arasında maliyet değerlendirmeleri yapılabilmiştir.

Yukarıda kısaca bahsedilen durum çalışmalarının ayrıntılarına bu bölümün ilerleyen kısımlarında yer verilmiştir.

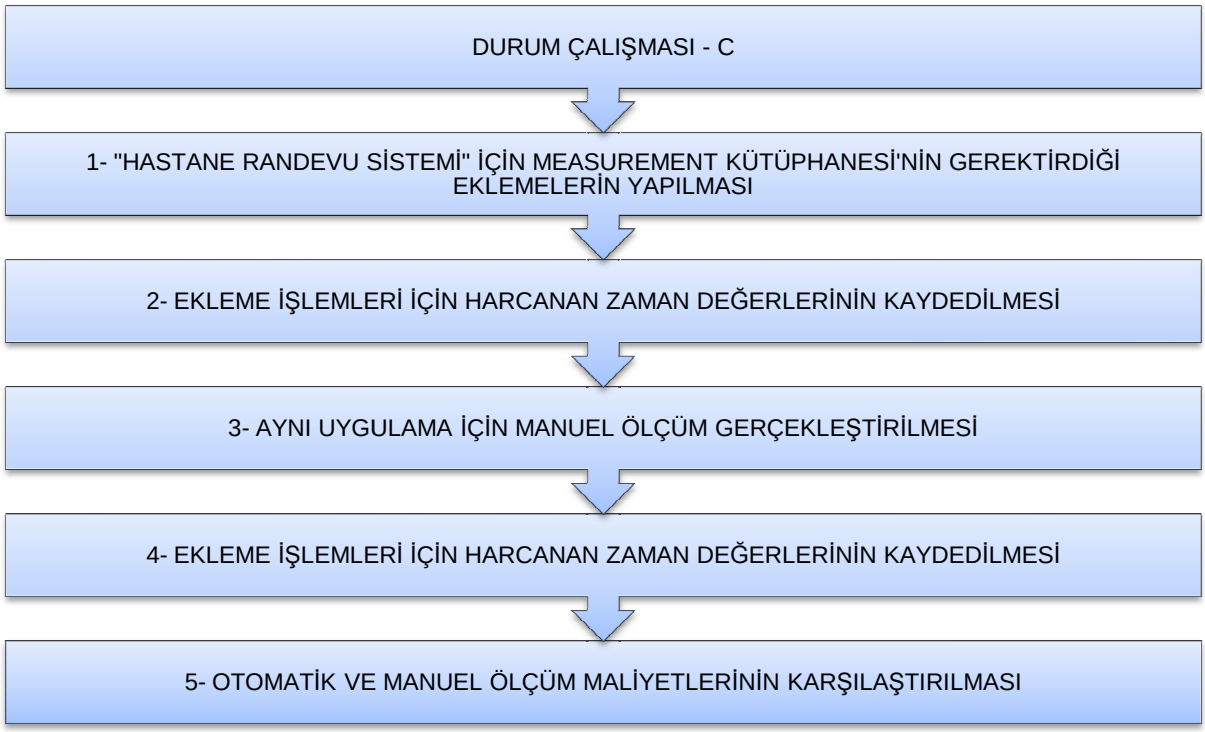
Durum çalışmalarına ilişkin olarak yapılan aksiyon planı aşağıda gösterilmiştir.



Şekil 6.3 Durum Çalışması A



Şekil 6.4 Durum Çalışması B



Şekil 6.5 Durum Çalışması C

6.2.2. Durum Çalışması A (İnternet Bankacılığı-1 Uygulaması - Yazılımcı-1)

Bu çalışma kapsamında, işlevsel büyüklük ölçme konusunda bilgisi olmayan bir yazılımcı (Yazılımcı-1) Measurement Kütüphanesi'ni gerçekleştirimini yaptığı "İnternet Bankacılığı-1" uygulaması içinden kullanmıştır. Gerçekleştirilen uygulama internet bankacılığına dair temel bazı işlevleri simüle etmekte olup, web üzerinde çalışmamaktadır. Uygulama Measurement Kütüphanesi gereksinimlerine uygun olarak Java programlama dili ile geliştirilmiş olup, üç-katmanlı mimariye sahiptir. Arayüz geliştirimi için "Swing" kütüphanesi kullanılmış olup, veri tabanı için MySQL seçilmiştir.

Yazılımcıya geliştirme öncesinde Measurement Kütüphanesi ile Ölçme Kılavuzu (bkz. Ek-2) temin edilmiş olup, geliştirim sırasında söz konusu kılavuzdan yararlanması sağlanmıştır. Yazılımcı uygulama geliştirimi ve Measurement Kütüphanesi'nin kullanımı kapsamında manuel olarak yapılan kod eklemeleri için harcanan zamanları ayrı ayrı kaydetmiştir. Gerçekleştirilen durum çalışması sırasında, kütüphanenin kullanılmaması durumunda katlanılacak manuel ölçme maliyetlerinin de değerlendirilebilmesi amacıyla, projenin farklı evrelerinde Yazılımcı-2 tarafından manuel ölçümler gerçekleştirilmiş olup harcanan süreler kaydedilmiştir.

Yapılan otomatik ve manuel ölçümler sonucunda elde edilen ve hesaplanan işlevsel büyüklüklere aşağıdaki tabloda yer verilmiştir.

Çizelge 6.3 İşlevsel Büyüklük Ölçme Sonuçları

DURUM ÇALIŞMASI-A	Otomatik Ölçüm (Yazılımcı - 1)	Manuel Ölçüm (Yazılımcı - 2)
İşlevsel Büyüklük (CFP)	39	39

Durum Çalışması A kapsamında Yazılımcı 1 tarafından uygulama geliştirme çalışmaları için kaydedilen toplam süre, Measurement Kütüphanesi'nin kullanımı için yapılan eklemeler için kaydedilen toplam süre ve Yazılımcı-2 tarafından yapılan manuel ölçme çalışmaları için kaydedilen toplam süre aşağıdaki tabloda özetlenmiştir.

Çizelge 6.4 Durum Çalışması A sırasında kaydedilen süreler

DURUM ÇALIŞMASI A	Uygulama Geliştirme Maliyeti	Measurement Kütüphanesini Kullanma Maliyeti	Manuel Yapılan Ölçüm Maliyeti*
ARA ÖLÇÜM -1	7 Saat 15 dakika	12 dakika	2 saat 10 dakika
ARA ÖLÇÜM - 2	13 saat 55 dakika	27 dakika	3 saat 15 dakika
TOPLAM SÜRE	17 Saat 25 dakika	40 dakika	4 saat 15 dakika

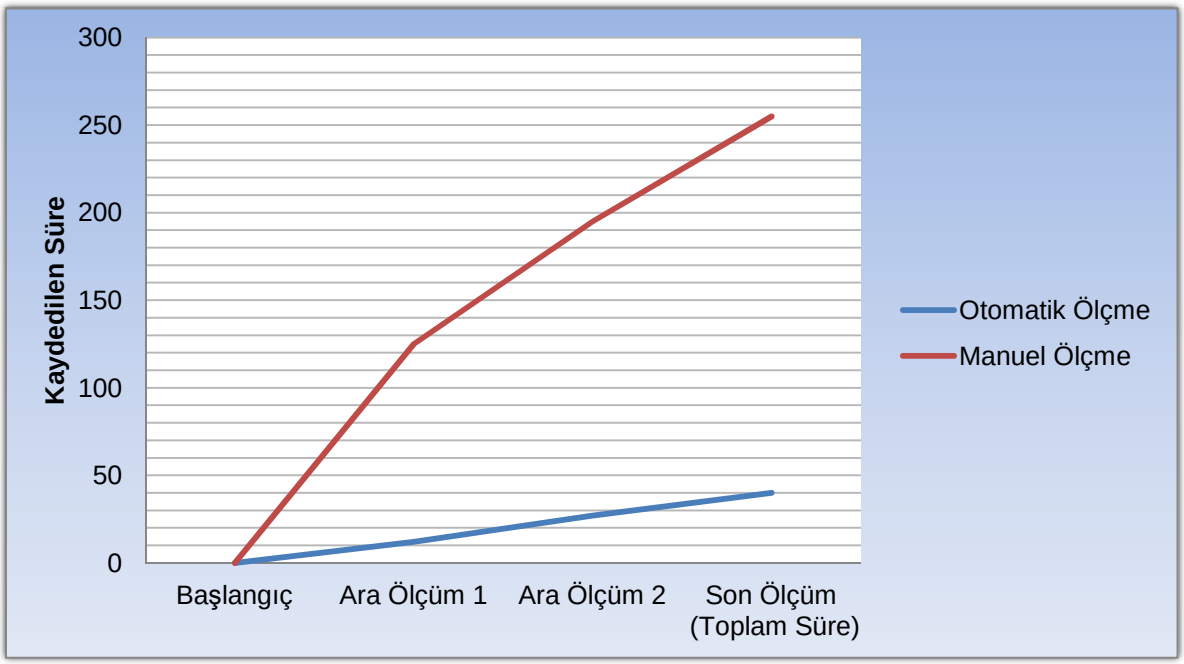
* Manuel yapılan ölçüm Yazılımcı 2 (Cosmic Uzmanı) tarafından gerçekleştirilmiştir.

Tabloda elde edilen sonuçlar incelendiğinde, İnternet Bankacılığı-1 uygulamasının Yazılımcı 1 için toplam geliştirme süresinin 17 saat 25 dakika olduğu, uygulama geliştirme süresince Measurement Kütüphanesi'nin gerektirdiği manuel kod ekleme işlemleri süresinin ise 40 dakika olduğu görülmektedir. Gerçekleştirilen uygulamanın Yazılımcı-2 tarafından manuel olarak yapılan işlevsel büyüklük ölçüm işlemi sırasında kaydedilen toplam süre ise 4 saat 15 dakikadır. Yazılımcı 1'in COSMIC işlevsel büyüklük ölçme metodu hakkında bilgi ve tecrübesinin olmaması sebebiyle manuel ölçüm işlemi Yazılımcı – 2 tarafından gerçekleştirilmiştir.

Elde edilen sonuçlar değerlendirildiğinde manuel yapılan ölçme işleminin, Measurement Kütüphanesi kullanılarak yapılan otomatik ölçme işleminden çok daha maliyetli olduğu anlaşılmaktadır. Yapılan ara ölçümler sırasında kaydedilen değerler, yazılım geliştirmenin her aşamasında manuel ölçme maliyetlerinin otomatik ölçme maliyetlerine göre oldukça yüksek olduğunu göstermektedir.

Manuel yapılan ölçümler sırasında yazılımın işleyişine hakim olmak gerektiğinden, yapılan ara ölçümlerin ilki diğer ölçümlere göre çok daha maliyetli olmuştur. İkinci ara ölçüm ve toplam süre ölçümü, Ara Ölçüm 1 sonrasında geliştirilen kısımlar değerlendirilerek yapıldığından görece olarak daha kısa sürmüştür. Ancak bu noktada manuel ölçümlerin aynı kişi tarafından gerçekleştirilmesi gerekmektedir. Aksi durumda ilk ölçümde katlanılan maliyete her ölçüm için katlanılması gerekecek ve manuel ölçüm maliyetleri katlanarak artacaktır. Measurement Kütüphanesi kullanıldığında ise koda doğru eklemeler yapıldığı sürece sonuçlar doğru ve maliyetler her zaman kabul edilebilir seviyelerde olacaktır.

Ara ölçümler ve nihai ölçüm sonunda elde edilen süre değerlerinin zamana göre değişimi aşağıdaki grafikte gösterilmiştir.



Şekil 6.6 Durum Çalışması A kapsamında yapılan ara ölçümler ve son ölçüm sırasında kaydedilen süre değerleri

Manuel ölçme işlemi sırasında ilk yapılan ölçme işleminde katlanılan maliyetin, uygulamaya aşina olduktan sonra, ikinci ve son adımda yapılan ölçümlerde azalarak

arttığı grafikten de görülmektedir. Otomatik yapılan ölçümde katlanılan maliyetler yazılımın ilerlemesi ile orantılı olarak artmaktadır.

Durum Çalışması A Measurement Kütüphanesinin, Kütüphane ve COSMIC işlevsel büyüklük ölçme konularında bilgi ve tecrübesi olmayan bir yazılımcı tarafından proje geliştirme sürecine entegre edilmesi ile başarılı sonuçların elde edilebileceğinin bir göstergesidir. Elde edilen veriler değerlendirildiğinde otomatik ölçmenin manuel ölçmeye göre %500'ü aşan oranlarda maliyetleri azalttığı görülmüştür.

Durum Çalışması A Measurement Kütüphanesi kullanılarak yapılan ölçme işleminin maliyet-etkin olduğunu göstermiştir. Durum Çalışması B İnternet Bankacılığı-1 uygulaması ile aynı işlevselliğe sahip uygulamanın Yazılımcı 2 tarafından geliştirilmesini ele almaktadır. Durum Çalışması B anlatıldıktan sonra, iki durum çalışması sonucunda elde edilen değerler karşılaştırılarak nihai değerlendirmeler yapılmıştır.

6.2.3. Durum Çalışması B (İnternet Bankacılığı-2 Uygulaması - Yazılımcı-2)

Durum Çalışması B kapsamında, işlevsel büyüklük ölçme konusunda bilgisi olan özellikle Cosmic işlevsel büyüklük ölçüm metodu konusunda tecrübeli bir yazılımcı (Yazılımcı-2) Durum Çalışması A kapsamında Yazılımcı 1 tarafından geliştirilen "İnternet Bankacılığı-1" uygulamasındaki işlevsellikle aynı işlevselliğe sahip "İnternet Bankacılığı-2" uygulaması geliştirmiştir. Durum Çalışması A ve B'den elde edilen sonuçların anlamlı olarak karşılaştırılabilmesi için aynı geliştirme aracı ve kütüphaneler kullanılarak geliştirilmiş olup, uygulamaların sağladıkları işlevselliklerin paralelliğine özellikle dikkat edilmiştir. Yazılımcı 2 de aynı şekilde Measurement Kütüphanesi'ni gerçekleştirimini yaptığı "İnternet Bankacılığı-2" uygulaması içinden kullanmıştır. Uygulama Measurement Kütüphanesi gereksinimlerine uygun mimariye ve altyapıya sahiptir.

Yazılımcı 2 uygulama geliştirimi ve "Measurement Kütüphanesi"nin kullanımı kapsamında yapılan manuel kod eklemeleri için harcanan zamanları ayrı ayrı kaydetmiştir. Gerçekleştirilen durum çalışması sırasında, Kütüphane'nin kullanılmaması durumunda katlanılacak manuel ölçme maliyetlerinin de değerlendirilebilmesi amacıyla, projenin farklı evrelerinde Yazılımcı-2 tarafından manuel ölçümler gerçekleştirilmiş olup harcanan süreler kaydedilmiştir.

Yapılan otomatik ve manuel ölçümler sonucunda elde edilen ve hesaplanan işlevsel büyüklüklere aşağıdaki tabloda yer verilmiştir.

Çizelge 6.5 İşlevsel Büyüklük Ölçme Sonuçları

DURUM ÇALIŞMASI-B	Otomatik Ölçüm (Yazılımcı - 2)	Manuel Ölçüm (Yazılımcı - 2)
İşlevsel Büyüklük (CFP)	39	39

Durum Çalışması B kapsamında Yazılımcı 2 tarafından uygulama geliştirme çalışmaları, Measurement Kütüphanesi için yapılan manuel kod ekleme işlemleri ve manuel ölçme çalışmaları için kaydedilen ara ölçüm ve toplam süre değerleri aşağıdaki tabloda özetlenmiştir.

Çizelge 6.6 Durum Çalışması B sırasında kaydedilen süreler

DURUM ÇALIŞMASI B	Uygulama Geliştirme Maliyeti	Measurement Kütüphanesini Kullanma Maliyeti	Manuel Yapılan Ölçüm Maliyeti*
ARA ÖLÇÜM -1	5 Saat 20 dakika	4 dakika	25 dakika
ARA ÖLÇÜM – 2	10 saat	15 dakika	45 dakika
TOPLAM SÜRE	14 Saat 30 dakika	25 dakika	1 saat 05 dakika

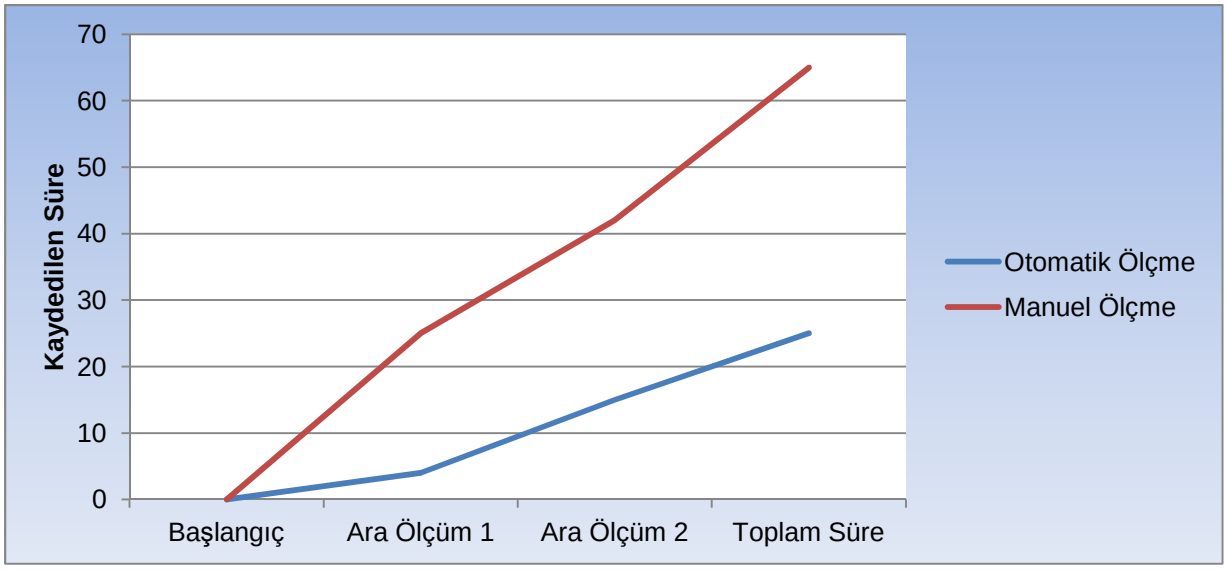
Tabloda elde edilen sonuçlar incelendiğinde, İnternet Bankacılığı-2 uygulamasının Yazılımcı 2 için toplam geliştirme süresinin 14 saat 30 dakika olduğu, uygulama geliştirme süresince Measurement Kütüphanesi'nin gerektirdiği ekleme işlemleri süresinin ise 25 dakika olduğu görülmektedir. Gerçekleştirilen uygulama için manuel olarak yapılan işlevsel büyüklük ölçüm işlemi sırasında kaydedilen toplam süre ise 1 saat 10 dakikadır.

Elde edilen sonuçlar değerlendirildiğinde Measurement Kütüphanesi'ne hakim ve COSMIC işlevsel büyüklük ölçümü konusunda deneyimli bir yazılımcının İnternet Bankacılığı-2 uygulaması geliştirimi sırasında yaptığı manuel ölçme işleminin,

Measurement Kütüphanesi'ni kullanılarak yaptığı otomatik ölçme işleminden daha maliyetli olduğu anlaşılmaktadır. Yapılan ara ölçümler sırasında kaydedilen değerler de manuel ölçme maliyetlerinin otomatik ölçme maliyetlerine göre yazılım geliştirme sürecinin her aşamasında daha yüksek olduğunu göstermektedir.

Durum Çalışması B için yapılan manuel ölçme işlemi, yazılımı geliştiren kişi tarafından gerçekleştirildiğinden Durum Çalışması A'da elde edilen değerlere göre daha küçük değerler elde edilmiştir. Ancak bu durumda dahi Measurement Kütüphanesi kullanılarak yapılan otomatik ölçme işleminin çok daha etkin olduğu anlaşılmaktadır.

Durum Çalışması B için gerçekeştirilen ara ölçümler ve nihai ölçüm sonunda elde edilen süre değerlerinin zamana göre değişimi aşağıdaki grafikte gösterilmiştir.



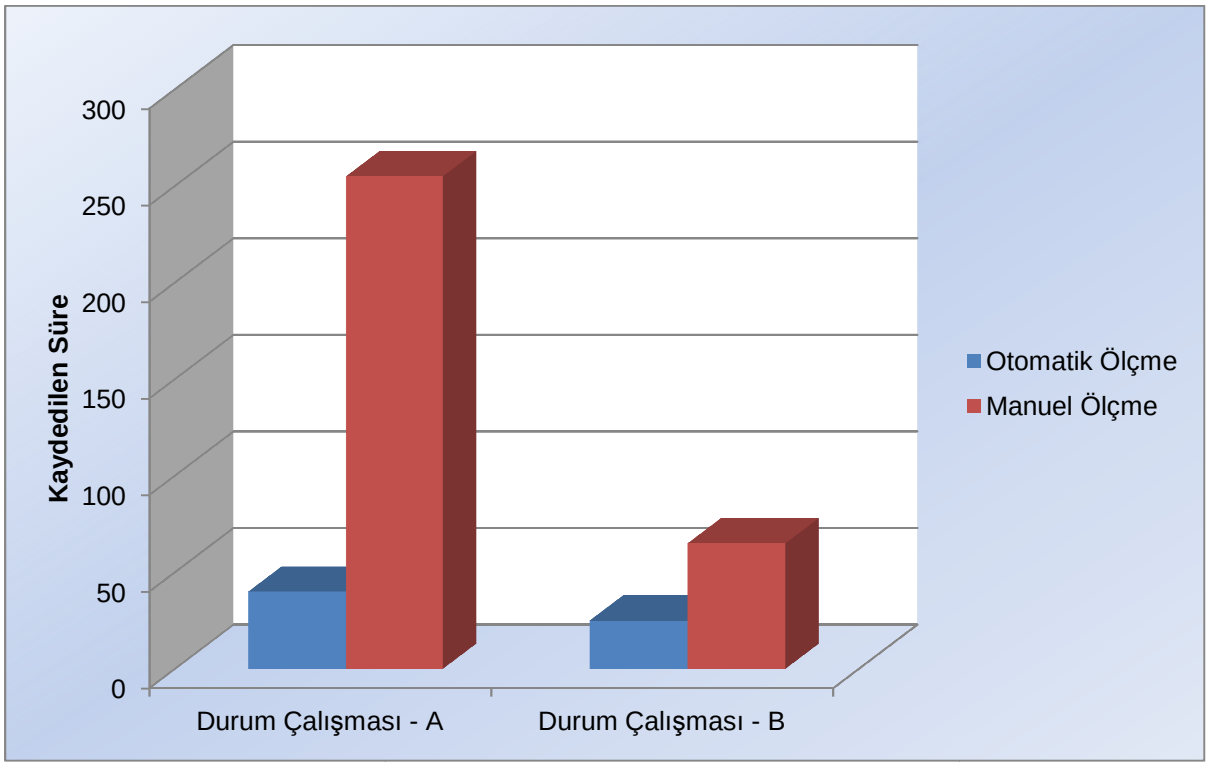
Şekil 6.7 Durum Çalışması B kapsamında yapılan ara ölçümler ve son ölçüm sırasında kaydedilen süre değerleri

Durum Çalışması B Measurement Kütüphanesinin proje geliştirme sürecine entegre edilmesi ile başarılı sonuçların elde edilebileceğinin ikinci göstergesi olmuştur. Manuel ölçmeye göre %250 oranında maliyet kazancı sağlamıştır. Durum Çalışması B kütüphane ve COSMIC metodu konularında tecrübeli bir yazılımcı tarafından Measurement Kütüphanesi kullanılarak yapılan otomatik ölçme işleminin manuel ölçmeye göre maliyet-etkin olduğunu göstermiştir.

Elde edilen sonuçlar Durum Çalışması A ile karşılaştırıldığında, kütüphanenin sağladığı maliyet avantajının daha düşük olduğu görülmüştür. Bu duruma sebep olan faktör, Durum Çalışması A için yapılan manuel ölçme işleminin, çalışma için

gerçekleştirilen İnternet Bankacılığı-1 uygulaması hakkında bilgisi olmayan Yazılımcı 2 tarafından gerçekleştirilmiş olmasıdır. Bu durumda manuel yapılan ölçüm daha uzun sürmüş, sağlanan kazanç bu nedenle daha yüksek olmuştur.

Durum Çalışması A ve Durum Çalışması B'den çıkarılabilecek ortak görüş, Measurement Kütüphanesi'nin yazılım geliştirme sürecine entegre edildiği durumlarda, yazılımcının kütüphane ya da COSMIC işlevsel büyüklük ölçme konularında herhangi bir bilgi ya da tecrübesi olmasından bağımsız olarak Measurement Kütüphanesi kullanılarak yapılan otomatik ölçme işleminin manuel yapılan ölçme işlemine göre maliyet-etkin olduğudur.



Şekil 6.8 Durum Çalışması A ve B kapsamında gerçekleştirilen çalışmalar sırasında kaydedilen süreler

6.2.4. Durum Çalışması C (Hastane Randevu Sistemi)

Durum Çalışması C kapsamında Hacettepe Üniversitesi BİL342 dersi kapsamında verilen Hastane Randevu Sistemi projesi ele alınmıştır. Measurement Kütüphanesi'nin kullanımı ile ilgili kısıtlara uyan bu proje üzerinde, gerekli eklemeler yapılarak otomatik işlevsel büyüklük hesaplanması amaçlanmıştır. Durum Çalışması C ile hedeflenen, proje geliştirme süreci tamamlanmış projeler için Measurement Kütüphanesi'nin kullanımıyla gerçekleştirilecek otomatik işlevsel büyüklük ölçümünün manuel yapılacak ölçüme göre maliyet-etkin olduğunun değerlendirilmesidir.

Öncelikle Masurement Kütüphanesi Ölçme Kılavuzunda (bkz. Ek-2) yer alan ekleme adımları uygulanmış ve bu sırada harcanan süre bilgisi kaydedilmiştir. Daha sonra aynı uygulama üzerinde manuel ölçme işlemi gerçekleştirilmiştir. Bu sırada harcanan süre değeri de kaydedilmiş ve otomatik yapılan ölçme işlemi sırasında kaydedilen değerlerle karşılaştırılarak bazı değerlendirmelerde bulunulmuştur. Bu değerlendirmelere bu bölümün sonunda yer verilmiştir.

"Measurement Kütüphanesi"nin kullanımı için gereken kod eklemeleri gerçekleştirecek olan yazılımcının Hastane Randevu Sistemi uygulamasının amacı, altyapısı, mimarisi ve nasıl geliştirildiği konularında bilgisinin bulunmaması, ekleme işlemlerinin yapılması için kaynak kodun neredeyse baştanbaşa analiz edilmesini gerektirmiştir. Arayüz üzerinde yer alan bileşenlerin hangisinin ne amaçla kullanıldığı, kod içerisinde herhangi bir olay tetikleyip tetiklemediği, veri tabanına erişip erişmediği, eğer eriştiyse hangi tablolara eriştiği ve hangi çıktıların üretildiği tek tek ele alınmış olup, daha sonra gerekli yerlere kütüphanenin kullanımına ilişkin kodlar eklenmiştir. Manuel ve kütüphanenin kullanılmasıyla otomatik olarak yapılan ölçme işlemleri sonucunda elde edilen işlevsel büyüklüklere tabloda yer verilmiştir.

Çizelge 6.7 İşlevsel Büyüklük Ölçme Sonuçları

Gerçekleştirilen Ölçüm	İşlevsel Büyüklük Otomatik Ölçme (CFP)	İşlevsel Büyüklük Manuel Ölçme (CFP)
Hastane Randevu Sistemi	46	46

Durum Çalışması C kapsamında Yazılımcı 2 tarafından geliştirme süreci tamamlanmış olan Hastane Randevu Sistemi uygulaması içinden Measurement

Kütüphanesi'nin kullanılması için yapılan manuel kod ekleme işlemleri ve manuel ölçme çalışmaları için kaydedilen süre değerleri aşağıdaki tabloda özetlenmiştir.

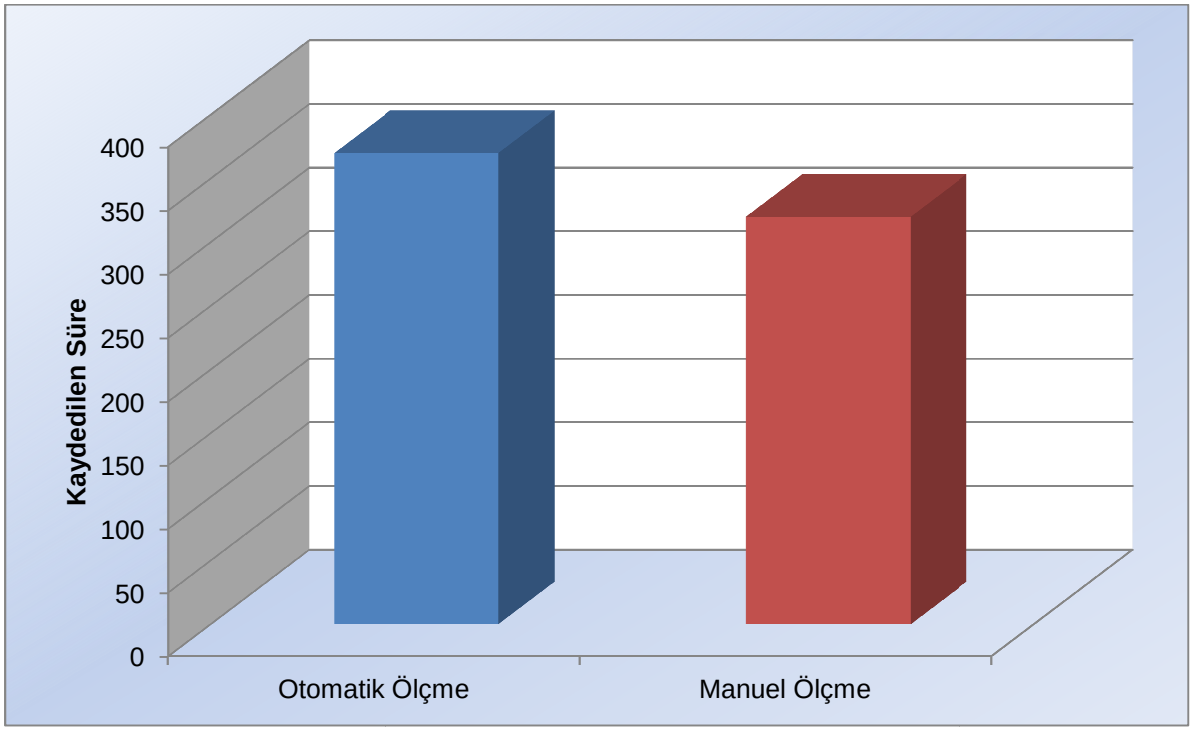
Çizelge 6.8 Durum Çalışması C sırasında kaydedilen süreler

Gerçekleştirilen Ölçüm	Uygulama Geliştirme Maliyeti	Measurement Kütüphanesini Kullanma Maliyeti	Manuel Yapılan Ölçüm Maliyeti*
Hastane Randevu Sistemi	Bilinmiyor	6 saat 10 dakika	5 saat 20 dakika

Tabloda yer alan değerler incelendiğinde, Hastane Randevu Sistemi uygulaması üzerine Measurement Kütüphanesi'ni kullanmanın gerektirdiği manuel kod ekleme işlemlerinin yapılma süresinin 6 saat 10 dakika olduğu, aynı uygulama için manuel olarak yapılan işlevsel büyüklük ölçüm işlemi sırasında kaydedilen toplam sürenin ise 5 saat 20 dakika olduğu tespit edilmiştir. Hastane Randevu Sistemi uygulaması geliştirme süreci Hacettepe Üniversitesi Bil342 dersi projesi kapsamında önceden tamamlandığından uygulama geliştirme maliyeti hakkında bilgi sağlanamamıştır.

Elde edilen sonuçlar değerlendirildiğinde, geliştirme süreci tamamlanmış uygulamalar için Measurement Kütüphanesi kullanılarak yapılan otomatik işlevsel büyüklük ölçümlerinin manuel yapılan ölçme işlemlerinden daha maliyetli olduğu anlaşılmıştır. Measurement Kütüphanesi'nin kullanımı için yapılması gereken manuel kod eklemeleri için doğru yerlerin belirlenmesi, kodun tamamının ayrıntılı olarak analiz edilmesini gerektirdiğinden bu sonuçlar elde edilmiştir. Yüksek ekleme maliyetlerine rağmen manuel ve otomatik ölçme işlemleri ile elde edilen sonuçların aynı olması Measurement Kütüphanesi'nin çalışma prensibinin doğruluğunu göstermektedir. Ancak otomatik ve manuel yapılan ölçümlerin maliyetleri göz önüne alındığında kütüphanenin kullanımı için uygulama tamamlandıktan sonra manuel olarak kod eklemenin uygun olmadığı anlaşılmıştır.

Durum Çalışması C sonunda otomatik ve manuel olarak gerçekleştirilen ölçme sonuçlarının karşılaştırılması tabloda verilmiştir.



Şekil 6.9 Durum Çalışması C Kapsamında gerçekleştirilen Otomatik ve Manuel ölçme işlemleri sırasında kaydedilen süreler

7. SONUÇLAR

Yazılım büyüklüklerinin işlevsel olarak ölçülmesi proje yönetimi açısından giderek önem kazanan bir konudur. Ancak işlevsel büyüklüklerin manuel olarak ölçülmesi projelerin bütünüyle analiz edilmesini gerektirdiğinden oldukça maliyetli bir süreçtir. Bu noktadan hareketle gerçekleştirilen Measurement Kütüphanesi ile işlevsel büyüklüklerin çalışma zamanında otomatik olarak ölçülmesini hedeflemiştir. Kütüphanenin kullanımı sırasında, işlevsel büyüklüğü ölçülecek olan yazılım üzerinde bazı kod parçalarının manuel olarak eklenmesi gerekmektedir. Gerekli eklemelerin yapılmasıyla birlikte projenin işlevsel büyüklüğü, uygulamanın çalışma zamanında işlevlerini birer kez çalıştırmak suretiyle ölçülebilmektedir.

Measurement Kütüphanesi'nin yazılım uygulamaları içinden kullanılması iki farklı aşamada gerçekleşebilir. Birincisi geliştirme sürecinin başından itibaren Measurement Kütüphanesi'nin kullanımının planlanması ve tüm aşamalar boyunca sağlanması, diğeri ise geliştirme sürecinin sonunda Measurement Kütüphanesi'nin uygulama içinden kullanımını sağlayacak kod eklemelerinin yapılmasıdır. Yapılan durum çalışmalarında her iki durum için farklı senaryolar üzerinden geliştirmeler yapılmış ve Measurement Kütüphanesi'nin uygulamalar içinden kullanımı sağlanmıştır. Bunun yanında her durum çalışmasındaki senaryo için manuel ölçme işlemi de gerçekleştirilmiş olup sonuçlar karşılaştırılmıştır. Karşılaştırmalar sonucu Measurement Kütüphanesi kullanılarak gerçekleştirilen otomatik ölçme işleminin hangi durumlarda maliyet-etkin olduğu hangi durumlarda maliyet-etkin olmadığı değerlendirilmiştir.

Measurement Kütüphanesi kullanımının yazılım geliştirmenin en başından itibaren planlanarak uygulamaya dahil edilmesi ile yazılım ürünü geliştirme boyunca izlenebilir, sürecin her adımında kazanılan işlevselliğin otomatik olarak hesaplanması ile projenin durumu hakkında değerlendirmeler yapılabilir, projenin kalan kısmının planlanması ve eğer proje planından sapmalar var ise projenin tekrar rayına oturtulması için gerekli aksiyonların alınması sağlanabilir.

Measurement Kütüphanesi'nin geliştirme sonunda uygulama içinden kullanıldığı durumda, projenin hedeflenen işlevselliğe sahip olup olmadığı değerlendirilebilir, buna göre katlanılan maliyetler gözden geçirilebilir.

Durum Çalışması A ve Durum Çalışması B kapsamlarında iki farklı profilde yazılımcı tarafından geliştirilen, aynı işlevselliğe sahip “İnternet Bankacılığı-1” ve “İnternet Bankacılığı-2” uygulamaları üzerinde, Measurement Kütüphanesi kullanımı üzerinde çalışılmıştır. Her iki durum çalışmasında da Measurement Kütüphanesi yazılım geliştirme sürecine en baştan katılmıştır. Durum Çalışması A Measurement Kütüphanesi ve COSMIC işlevsel büyüklük ölçme metodu konularında bilgi sahibi olmayan bir yazılımcı tarafından gerçekleştirilmiştir. Durum Çalışması B ise Measurement Kütüphanesi ve Cosmic işlevsel büyüklük ölçme metodu konularında tecrübeli bir yazılımcı tarafından gerçekleştirilmiştir.

Durum Çalışması A ve B için elde edilen sonuçlar değerlendirildiğinde Measurement Kütüphanesi kullanımının yazılım geliştirme sürecine en baştan ve doğrudan entegre edildiği durumlarda başarılı olduğunu ortaya konmuştur. Yapılan uygulamada elde edilen değerler neticesinde %500’ü aşan işgücü kazanımlarının gerçekleştiği belirlenmiştir. Buradan hareketle, Measurement Kütüphanesi’nin yazılım geliştirme sürecine en başından adapte edildiği ve projenin, kütüphanenin kullanımı için gerekli kod eklemeleri yaparak ilerlediği durumlarda; elde edilen sonuçlar, kütüphanenin işlevsel büyüklüğün otomatik olarak ölçülmesi amacıyla kullanılmasının maliyet-etkin olduğunu kanıtlamıştır.

Durum Çalışması C ise geliştirme süreci tamamlanmış bir yazılım projesi üzerine Measurement Kütüphanesi’nin entegre edilmesini ele almıştır. Kütüphane’yi kullanmanın gerektirdiği kod eklemeleri ve manuel olarak yapılan işlevsel büyüklük ölçme işlemi için harcanan süreler kaydedilmiş ve elde edilen sonuçlar karşılaştırılarak değerlendirmeler yapılmıştır.

Yapılan değerlendirmelerde Measurement Kütüphanesi’nin başarılı olacağı düşünülen diğer bir nokta olan uygulama geliştirimi sonrası Measurement Kütüphanesi entegrasyonu işleminin beklenen başarıyı göstermediği anlaşılmıştır. Kütüphane kullanılarak gerçekleştirilen ölçme işlemi, ekleme işlemlerini gerçekleştiren yazılımcının kodun mimarisi ve işlevi hakkında bilgisi olmaması sebebiyle kodun bütününün analizini gerektirmiştir. Bu nedenle manuel ölçme işleminde yapılan neredeyse tüm adımlar kod ekleme çalışmaları için gerçekleştirilmek zorunda kalmıştır. Sonuç olarak kütüphanenin tamamlanmış uygulamalar için, uygulamaya yabancı bir kimse tarafından entegre edilmesinin süreci uzattığı görülmüş olup, manuel ölçmeye göre bir avantaj sağlanamamıştır. Bu

durumda Measurement Kütüphanesinin otomatik ölçme için uygulamaya entegre edilmesi maliyet-etkin değildir.

Gelecek çalışmalarda Measurement Kütüphanesi'ni kullanmanın gerektirdiği kod ekleme işlemlerinin ve çalışma zamanında kullanıcı arayüzü üzerinden veri hareketi tetikleme eylemlerinin otomatize edilmesi hedeflenmektedir. Eklemelerin, üzerinde uygulanacak yazılımın otomatik olarak analiz edilmesi neticesinde uygun yerlere yapılabileceği, bu sayede tamamlanmış bir proje için yapılan ekleme işlemleri sonucunda gerçekleştirilen otomatik ölçme işleminin maliyet-etkin olacağı düşünülmektedir. Measurement Kütüphanesi'nin geliştirilmesi için farklı teknolojiler kullanılarak geliştirilen yazılımların işlevsel büyüklüklerinin de otomatik olarak ölçülebilmesi için gereken araştırma ve çalışmalar yapılabilir. Ayrıca diğer işlevsel büyüklük ölçme metodları için de otomatikleştirme çalışmalarının uygulanabilirliği konusunda çalışmalar gerçekleştirilebilir.

8. REFERANSLAR

[1] Morasca S.: Software Measurement, Handbook of Software Engineering and Knowledge Engineering, (S.K. Chang, ed.), Chapter 2: Software Measurement, World Scientific, pp. 239-276, 2001.

[2] Internet: “The Standish Group Report”, http://www.standishgroup.com/, 2011.

[3] Fenton, N.E., ve Pfleeger, S.L., Software Metrics: A Rigorous and Practical Approach (2nd Ed.). PWS Publishing Company, 1997.

[4] Kurtel K., Eren Ş., “Yazılım Ölçümü: Genel Bir Bakış,” Yazılım Kalitesi ve Yazılım Geliştirme Araçları Sempozyumu (YKGS), 2008.

[5] Anbari F. T. , “Earned value project management method and extensions,” Project Management Journal 34(4) p.12-23, December 2003.

[6] Brandon D. M., “Implementing earned value easily and effectively,” Project Management Journal 2(29) p.11-18, June 1998.

[7] Common Software Measurement International Consortium, “COSMIC Method Version 3.0.1, Measurement Manual,” 2009.

[8] International Function Points Users Group, “Function point counting practices manual, Release 4.3,” 2010.

[9] Minkiewicz A. F., “The evolution of software size: A search for value,” CROSSTALK The Journal of Defense Software Engineering, Vol. 22, No. 3 p.23-26, March/April 2009.

[10] Cuadrado J. J., Crespo J., Sicilia M. A., A. de Amescua and Garcia L., “Comparative analysis of different methods of function points measurement”, Software Measurement European Forum (SMEF), Rome, Italy 2004.

[11] Total Metrics, “How to decide which method to use,” http://www.totalmetrics.com/function-point-resources/downloads/R185_Why-use-Function-Points.pdf, Version 1.1, August 2007.

[12] Ho V. T. and Abran A., “A Framework for automatic function point counting from source code”, International Workshop on Software Measurement (IWSM’99), September 8-10, 1999.

[13] Jones E. L., “Automated calculation of function points,” Proceedings of the 4th Software Engineering Research Forum, SERF, Boca Raton, Florida, 1995.

[14] Kusumoto S. et al., “Function point measurement from Java programs,” proceedings of the 24th International Conference on Software Engineering (ICSE2002) p.576-582, 2002.

[15] Diab H., Koukane F., Frappier M. and St-Denis R., “µcROSE: Automated Measurement of COSMIC-FFP for Rartional Rose RealTime”, Elsevier Science 29 June 2004

[16] Paton K., “Automatic function point counting using ststic and dynamic code analysis,” International Workshop on Software Measurement, p.6, IWSM’99.

[17] International Standarts Organization and Inernational Electrotechnical Commission, “Mk II function point analysis,” Counting Practices Manual, ISO/IEC 20968:2002.

[18] International Standarts Organization and Inernational Electrotechnical Commission, “IFPUG functional size measurement method,” ISO/IEC 20926:2009.

[19] International Standarts Organization and Inernational Electrotechnical Commission, “COSMIC: a functional size measurement method,” ISO/IEC 19761:2011.

[20] International Standarts Organization and Inernational Electrotechnical Commission, “NESMA functional size measurement method version 2.1,” Definitions and counting guidelines for the application of Function Point Analysis, ISO/IEC 24570:2005.

[21] International Standarts Organization and Inernational Electrotechnical Commission, “FiSMA 1.1 functional size measurement method,” ISO/IEC 29881:2010.

[22] Common Software Measurement International Consortium, “The COSMIC Functional Size Measurement Method Version 3.0, Guideline for Sizing Business Application Software Version 1.1,” May, 2008.

[23] Abran A., “FFP Version 2,0: An implementation of COSMIC functional size measurement concepts,” Keynote presentation at the FESMA 99 Conference, Amsterdam, Oct. 7, 1999

[24] Abran A., Meli R., and Symons C., “COSMIC-FFP (ISO 19761) software size measurment: State of the Art,” Software Measurement European Forum (SMEF2004), Rome, Italy, January 28-30 2004.

[25] Top O O. Demirörs O, and Ozkan B. “Reliability of cosmic functional size measurement results: A multiple case study on industry cases,” 35th Euromicro Conference on Software Engineering and Advanced Applications, p.327-334, Greece, 27-29 August 2009

[26] Lavazza L, and Bianco V. D., “A case study in Cosmic functional size measurement: The rice cooker revisited,” IWSM/Mensura 2009, LNCS 5891, pp. 101–121, Amsterdam, Netherlands, 2009.

[27] Fenton N. E. and Pfleeger S. L., Software Metrics “A Rigorous & Practical Approach”, Second Edition, PWS Publishing Company, 1997

[28] IEEE, “IEEE Standart Glossary of Software Engineering Terminology”, ISBN 1-55937-067, September 28, 1990

[29] Abran A., “Software Metrics and Software Metrology”, IEEE computer society and A John Wiley & Sons, Inc., Publication, 2010

[30] Conte S. D., Dunsmore H. E., Shen V. Y., “Software Engineering metrics and models”, Benjamin/Cummings Pub. Co., March 1986

[31] Kan S. H., “Metrics and Models in Software Quality Engineering, Second Edition”, Pearson Education, Inc., July 2004

[32] Galorath D. D. and Evans M. W., “Software sizing, estimation, and risk management : when performance is measured performance improves.” Auerbach Publications, 2006

[33] Boehm B. W. and Sullivan K. J., “Software Economics: A Roadmap”, Future of Sofware Engineering, ACM, Limerick-Ireland, 2000

[34] Grady R. B. and Caswell D. L., Software Metrics: Establishing a Company-Wide Program, Prentice-Hall, 1987

[35] Pfleeger S. L., An investigation of cost and productivity for object-oriented development, George Mason University Press Faifax, VA, USA, 1989

[36] Lorenz M., Object-oriented software development: a practical guide, Prentice-Hall, Inc., NJ, USA, 1993

[37] Krueger C. W., Software Reuse, ACM Computing Surveys, Vol.24, No.2, p131-183, Pittsburgh, Pennsylvania, June 1992

[38] McIlroy M. D., Mass Produced Software Components, NATO Software Engineering Conference Report, p.138,151, Garmisch, Germany – 7-11 October 1968

[39] Barnes B. and Bollınger T., Making software reuse cost effective, IEEE Software vol.8 p.13-24, 1991

[40] Jones C., Software return on investment preliminary analysis, Software Productivity Research, Inc., 1993

[41] Demarco T.. Controlling Software Projects, Yourdon press, New York, 1982.

[42] Boehm, Barry B., and Fairley, Richard E., Software Estimation Perspectives, IEEE Software November-December, pp.22-26, December 2000

[43] Longstreet, D., Fundamentals of Function Point Analysis, Longstreet Consulting Inc., http://www.softwaremetrics.com/fpafund.htm

[44] Levesque G., Bevo V. and Cao D. T., “Estimating software size with UML models, Proceedings of the 2008 C3S2E Conference(C3S2E-08), Montreal, Canada, May 12-13 2008.

[45] Boehm B. et al., Cost Models for Future Software Life Cycle Processes: COCOMO 2.0, Annals of Software Engineering Special Volume on Software

Process and Product Measurement J.D. Arthur and S.M. Henry, eds., vol. 1, pp. 45-60, J.C. Baltzer AG, Science, Amsterdam, The Netherlands, 1995

[46] Evans M. W., Marciniak J. Software Quality Assurance and Management, John Wiley and Sons, 1987.

[47] Whitmire S. A. Object - Oriented Design Measurement, John Wiley and Sons, 1997

[48] Internet: Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Project_management

[49] Guidelines for Successful Acquisition and Management of Software-Intensive Systems:Weapon Systems Command and Control Systems Management Information Systems Version 3.0, Department Of The Aır Force Software Technology Support Center, May 2000.

[50] Ishigaki D. and Jones C., Practical Measurement in the Rational Unified Process, http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jan03/PracticalMeasurementInRUP_TheRationalEdge_Jan2003.pdf, January 2003

[51] Internet: Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Earned_value_management

[52] Wilkens T. T., Earned Value, Clear and Simple, http://www.projectsmart.co.uk/docs/earned-value.pdf, April 1, 1999

[53] Internet: MS Project 2003 user guide, About Earned Value Analysis., http://office.microsoft.com/en-001/project-help/about-earned-value-analysis-HP001034258.aspx

[54] İnternet: NASA, Earned Value Management “What is EVM?”, http://evm.nasa.gov/

[55] USA Department of Defense, Earned Value Management Implementation Guide, October, 2006

[56] Project Management Institute Inc, “A Guide to the Project Management Body of Knowledge”, 2008 (An American National Standard ANSI/PMI 99-001-2008)

[57] Cabri A., Griffiths M., Earned Value and Agile Reporting, Agile Conference, p.6-22, 2006

[58] Common Software Measurement International Consortium, “The COSMIC Functional Size Measurement Method Version 3.0, Advanced and Related Topics” Dec, 2007.,

[59] R. Rask, “Algorithms for counting unadjusted function points from dataflow diagrams,” University of Joensuu Department of Computer Science Report Series A-1991-1.

[60] F. Gramantieri, E. Lamma, P. Mello and F. Riguzzi, “A system for measuring function points from specifications,” Technical Report DEIS-LIA-97-006., 1997

[61] G. Levesque, V. Bevo and D. T. Cao, “Estimating software size with UML models, Proceedings of C3S2E conference (C3S2E-08), Montreal/Canada, 12-13 May 2008.

[62] Jenner M.S., COSMIC-FFP and UML: Estimation of the Size of to System Specified in UML-Problems of Granularity. In Proc. Fourth European Conf. Soft. Measurement and ICTWith-trol p.173-184., Germany,May 2001.

[63] Poels, G. A Functional Size Measurement Method for Event-Based Object-Oriented Enterprise Models. Proceedings of the 4th International Conference on Enterprise Information Systems–ICEIS, p. 667–675, Ciudad Real, Spain, 2002.

[64] Azzouz, S. & Abran, A. A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: *COSMIC FFP*. Proceedings of the Software Measurement European Forum (SMEF2004), Rome , Italy, 2004.

EK-1: “MEASUREMENT KÜTÜPHANESİ” Kullanım Kılavuzu

“MEASUREMENT KÜTÜPHANESİ”

Kullanım Kılavuzu

GİRİŞ

Ölçme Kavramı

Ölçme, günlük hayat ile iç içe geçmiş, tüm bilimsel ve mühendislik disiplinleri için olmazsa olmaz bir kavramdır. Herhangi bir konuda yapılan bir çalışmada değerlendirme yapabilmek için, ölçme işlemine ihtiyaç duyarız. İnşası yapılan bir evin büyüklüğü, içinde bulunan ortamın sıcaklığı, herhangi bir proje için harcanan zaman, ancak ölçme işlemi sayesinde belirlenebilir. Elde edilen sonuçlar değerlendirilerek sonraki çalışmalar için kaynak oluşturulabilir, içinde bulunan durumun analizi yapılabilir ve önceki çalışmalar için değerlendirmede bulunulabilir.

Yazılımların büyüklüklerinin ölçülmesi, yazılım mühendisliği disiplini için önemi hızla artan bir konudur. Büyüklüğün yazılım geliştirmenin ilk adımlarından itibaren ölçülebilir olması, ölçme işlemini artan yazılım boyutları nedeniyle kritik önem kazanan yazılım proje yönetimi sürecinin en önemli bileşenlerinden biri haline getirmiştir. Erken evrelerde ölçme, proje planlamasının çok daha etkin bir şekilde gerçekleştirilebilmesini; bu sayede projenin belirlenen takvim içerisinde, hangi kaynakların kullanımıyla ve ne kadar maliyetle gerçekleştirilebileceğinin önceden belirlenebilmesini sağlar. Geliştirme sırasında yapılan ölçümler, planlanan yazılım büyüklükleri ile karşılaştırılarak projenin gelişiminin yorumlanmasında kullanılabilir. Yazılım süreçlerinin yönetimi, geliştirilen yazılım üzerinde iyileştirmelerin yapılması, yazılım geliştirme sürecinde harcanan işgücü ve katlanılan maliyet değerlendirmelerinin yapılması, ölçme sayesinde gerçekleştirilmektedir. Yazılım mühendisliğinde ölçme, yazılım süreçlerinin ve ürünlerinin anlaşılması, kontrol edilmesi ve mevcut başarımlarının izlenerek iyileştirilmesi amacıyla uygulanan etkinliklerdir.

Yazılım büyüklüklerinin ölçülmesi manuel olarak gerçekleştirildiğinde oldukça zaman alıcı ve maliyetli bir süreç olmaktadır. Measurement Kütüphanesi bu noktadan hareketle geliştirilmiş, proje geliştirimine doğrudan entegre edilebilen ya da proje geliştirimi sonrası ölçme amaçlı koda dahil edilebilen bir araçtır. Kullanımı sırasında

dikkat edilecek temel hususlar bu belgede anlatılmıştır. İlerleyen bölümlerde Measurement Kütüphanesi'nin genel yapısı, kullanım adımları, geliştirime açıklığı ve örnek uygulama üzerinde nasıl kullanıldığına yer verilecektir.

Neden Measurement Kütüphanesi?

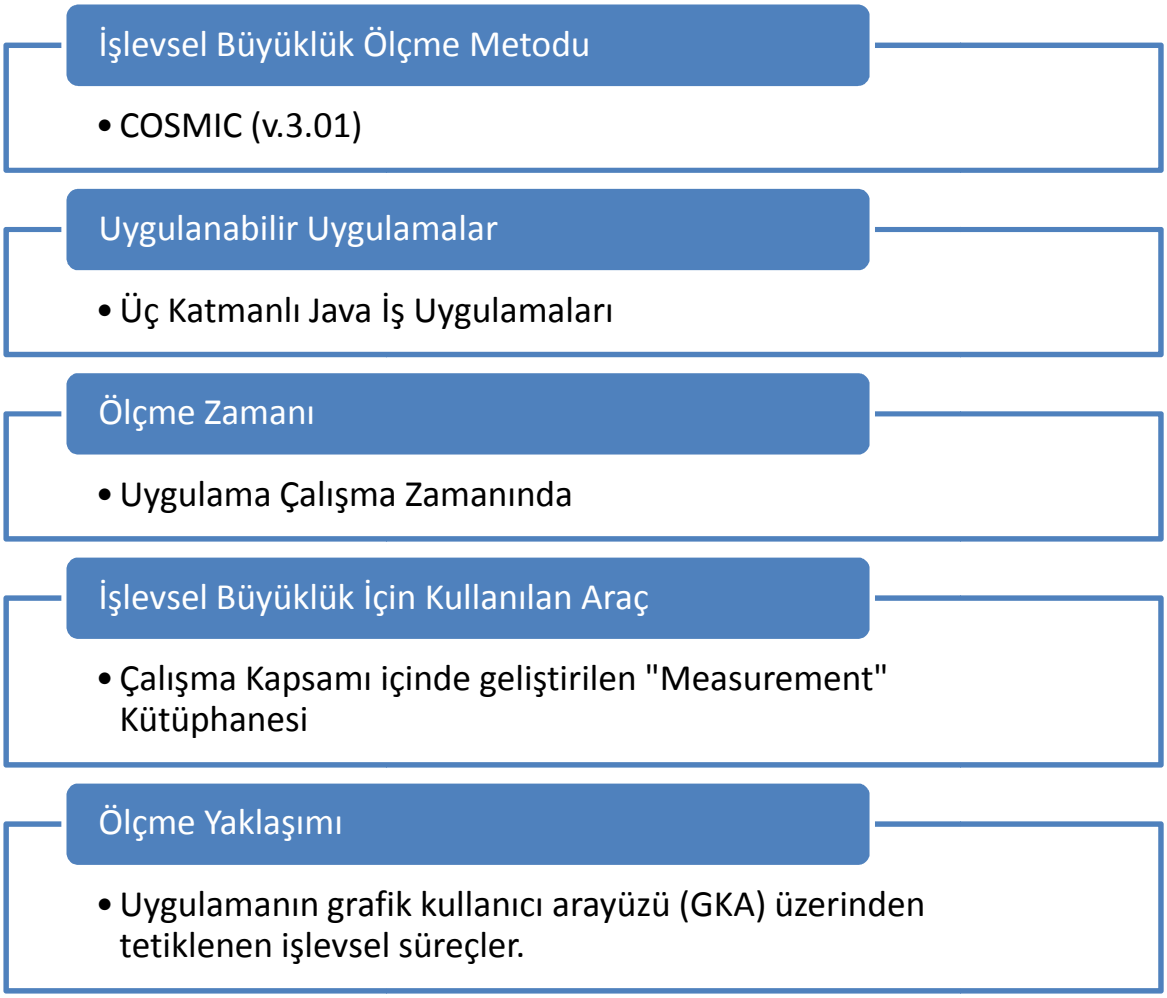
Giriş kısmında bahsedilen kritikliği sebebiyle ölçme işlemi proje yönetimi için olmazsa olmaz niteliktedir. Ölçme işleminin proje geliştirme süreci devam ederken her adımda yapılabilir olması, projenin mevcut durumu konusunda anlık bilgi sağlanabilmesi için oldukça önemlidir. Her adımda manuel olarak bu işlemi gerçekleştirmek maliyet açısından etkin bir yöntem olmayacaktır. Bu noktada çözüm ölçmenin otomatik olarak geçekleştirilebilmesidir. İşte tam da bu noktada Measurement Kütüphanesi devreye girer. Proje geliştirimine entegre edilmesiyle birlikte, projenin anlık olarak işlevsel büyüklüğünün hesaplanmasını sağlar. Önceden belirlenen işlevselliklere göre projenin cari durumu için değerlendirmeler yapılması sağlanırken, proje yönetiminde bir sonraki hamlenin ne olacağının kararının da verilmesi konusunda yol gösterici olur.

Measurement Kütüphanesi Kullanım Kısıtları

Measurement Kütüphanesi uygulamanın işlevsel büyüklüğünün ölçülmesi için tasarlanmıştır. İşlevsel büyüklük ölçme için COSMIC metodu temel alınmış olup, yapılan ölçümler bu ölçme metodunun kısıtları çerçevesinde gerçekleştirilmektedir. Farklı bir yöntem kullanılarak elde edilen ölçme sonuçlarıyla karşılaştırılması anlamlı sonuçlar vermeyecektir.

Kütüphane Java ile geliştirilmiş üç katmanlı mimariye sahip uygulamalar için tasarlanmıştır. GKA ve veri tabanı hareketleri ölçmenin temelini oluşturmaktadır.

Measurement Kütüphanesi'nin temelleri şekilde gösterilmiştir.



İlk olarak, ölçme yöntemi ile sadece GKA (Grafik Kullanıcı Arayüzü) üzerinden tetiklenen işlevselliğin ölçümü otomatik olarak yapılabilir. Uygulamanın iletişimde

olduğu diğer uygulamalar varsa ve bunlar tarafından tetiklenen işlevler GKA üzerinden çağrılmıyorsa bu işlevlerin büyüklüğü bu yöntemle hesaplanamaz. Benzer şekilde eğer yapılan işlemler bir veri tabanı üzerinde gerçekleşmiyor, farklı bir şekilde kaydediliyor ise (xml file, etc.) Measurement Kütüphanesi kullanılarak doğru sonuçlar almak mümkün olmayacaktır.

İkinci olarak, Measurement Kütüphanesi GKA üzerindeki bileşenler aracılığıyla tetiklenen eylemleri dikkate almaktadır. Tetikleyici bileşenlerin türü veya sayısı "Measurement" kütüphanesinin işlevselliğini değiştirmemekte, sadece uygulama içine eklenecek kodu etkilemektedir.

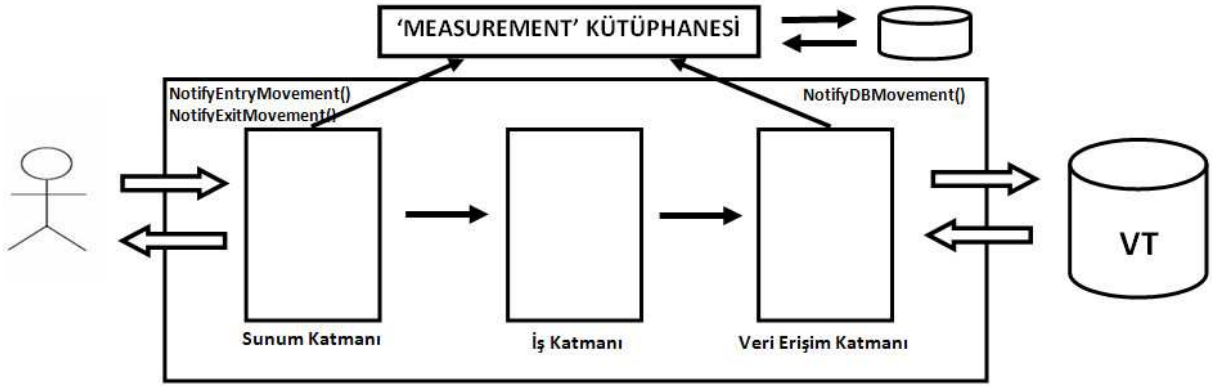
Üçüncü olarak, daha önce de belirtildiği üzere, büyüklüğü hesaplanacak işlevselliğin uygulamanın çalışması sırasında GKA üzerinde en az bir kez işletilmesi gerekmektedir. GKA üzerinde bir kez işletilmeyen işlevlerin büyüklüğü, yöntemle hesaplanamaz.

Dördüncü olarak, GKA üzerinden tetiklenen eylemlerin başlattığı hareketlerin bir işlevsel süreç oluşturup oluşturmadığı, diğer bir deyişle işlevsel süreç keşfi, hareketlerin zaman bilgisini kullanarak yapılmaktadır. Kütüphanenin keşif operasyonunda bu zaman aralığı 1 saniye olarak belirlenmiştir. Ne var ki üç mantıksal katmanlı uygulamanın fiziksel mimarisi, veri hareketlerinin gerçekleşme zamanı üzerinde etkili olabilir. Bu durumda işlevsel süreçler yanlış keşfedilebilir.

Beşinci olarak kütüphanenin kullanılması için yapılacak kod eklemeleri kullanım kılavuzunda yer alan yönlendirmeler doğrultusunda eklenmelidir. Yanlış ya da eksik yapılan eklemeler neticesinde hatalı sonuçlar elde edilebilir. Bu konuya özellikle dikkat etmek gereklidir.

Measurement Kütüphanesi ile Otomatik Ölçmenin Genel Mimarisi

Measurement Kütüphanesi üç katmanlı Java uygulamaları için uygulama geliştirme sürecine entegre olabilen bir araçtır. Kütüphane ile gerçekleştirilen otomatik ölçme metodunun mimarisi şekilde verilmiştir.



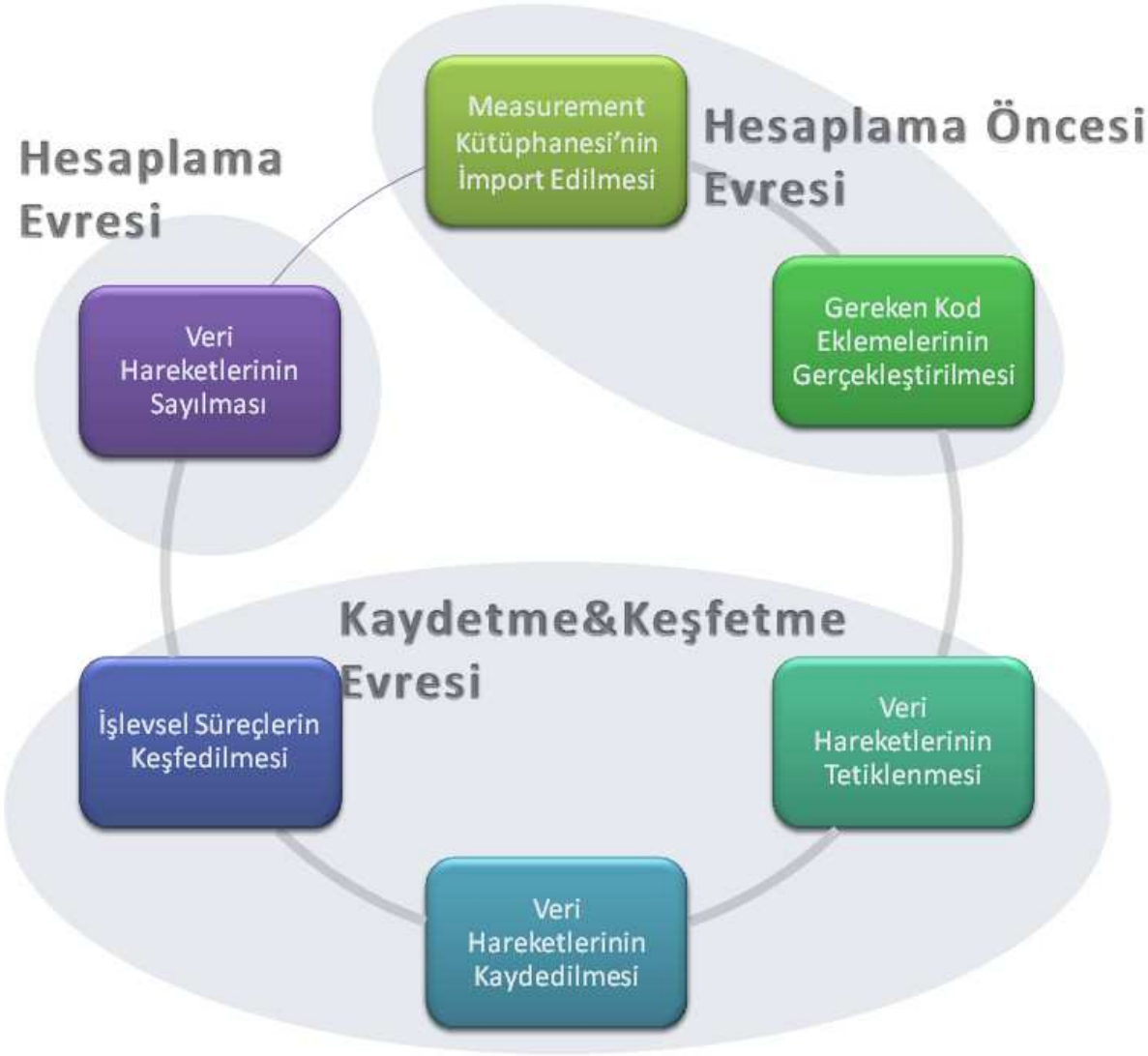
Şekilde görüldüğü üzere, Measurement Kütüphanesi'nin uygulamanın katmanları içerisinde tanımlanmış olması gereklidir. Ayrıca Kütüphane'nin metotları ilgili katmandaki veri hareketine uygun olarak koda yerleştirilmelidir.

Kütüphaneye ait NotifyEntryMovement() ve NotifyExitMovement() metotları kullanıcı ile etkileşim içinde olan katmana eklenirken, NotifyDBMovement() metodu veri tabanı hareketlerinin olduğu katmana eklenmelidir.

Measurement Kütüphanesi ile Ölçme

Measurement Kütüphanesi ile işlevsel büyüklük ölçümü üç temel evrede altı farklı adımda gerçekleştirilebilmektedir. Bu evre ve adımlar şunlardır:

1- Hesaplama Öncesi Evre
   a. Measurement Kütüphanesi'nin import edilmesi
   b. Gerekli kod eklemelerinin yapılması
2- Kaydetme ve Değerlendirme Evresi
   a. Veri Hareketlerinin Tetiklenmesi
   b. Veri Hareketlerinin Kaydedilmesi
   c. İşlevsel Süreç Keşfi
3- Hesaplama Evresi
   a. Veri Hareketlerinin Sayılması



1- Hesaplama Öncesi Evre

a) Measurement Kütüphanesi'nin import edilmesi

Measurement Kütüphanesinin uygulama yazılımı içerisinde kullanılabilmesi için öncelikle kütüphane paketinin kod içerisinden import edilmesi gerekmektedir. Java dilinde "import" anahtar sözcüğüyle gerçekleştirilen bu işlem ilgili sınıfın tanımında gerçekleştirilir. Örnek kullanıma aşağıda yer verilmiştir.

```
import dizin.MeasurementLibrary;
```

b) Gerekli Kod Eklemelerinin Yapılması

Kütüphanenin kod içerisinden import edilmesi sonrasında gerekli kod eklemelerinin yapılması evresine geçilir. Measurement Kütüphanesi kullanılarak yapılan otomatik ölçme işleminin en kritik evresidir. Bu evrede yapılacak kod eklemeleri sonraki adımlarda elde edilecek değerler için temel niteliği taşımaktadır.

Mesurement Kütüphanesi'nin uygulama geliştirme sürecine entegre edilmediği, geliştirme sonrası uygulandığı durumlarda, oldukça maliyetli bir evredir. Eklemeleri yapan kişinin koda olan aşinalığına göre süreç uzayıp kısalabilir. Kod hakkında hiçbir bilginin olmadığı durumlarda yapılacak kod eklemelerinde otomatik ölçme işleminin hedeflenen iş gücü kazanımını sağlayamadığı yapılan çalışmalarla tespit edilmiştir.

Burada önemli diğer bir husus ise arayüz üzerindeki bileşenlere ait dinleyicilerin ("listener") tanımlanmış olması gerekmektedir. Daha önce de anlatıldığı üzere Measurement Kütüphanesi arayüz üzerinden tetiklenen eylemleri ölçme için dikkate almaktadır. Bileşenler ve tanımlı olması beklenen dinleyiciler aşağıdaki listede belirtilmiştir:

Bileşen (Java Swing Component)	Tanımlı olması gereken dinleyici (Java Listener)
JButton	***ActionListener***
JTextField	***PropertyChangeListener, DocumentListener***
JComboBox	***ActionListener***
JMenuItem	***ActionListener***
JList	***PropertyChangeListener***
JDialog	***WindowsListener***
JFrame	***WindowListener***
JTabbedPane	***ChangeListener***
JTable	***PropertyChangeListener***

Gerekli eklemeleri yapmadan önce Measurement Kütüphanesi'nin, koda ekleyeceğimiz metotlarını inceleyelim. Kütüphane 4 adet metot içermektedir. Bu metotlar COSMIC ölçme yönteminin temel aldığı 4 temel veri hareketi ve değerlendirme sonrası toplam işlevsel büyüklüğün hesaplanması için tanımlanmıştır.

NotifyEntryMovement(): GKU üzerindeki bileşenler için tanımlı hareket dinleyiciler tarafından Giriş ("Entry") hareketlerinin algılanması için çağrılır. İlgili olayı parametre olarak alır ve buradan yola çıkarak tetikleyici kimliğine ulaşır. Tetikleyici kimlikleri saklanarak ölçme sırasında aynı kaynaktan gelen birden fazla hareket için mükerrer kayıt yapılmasının önüne geçilir. Yine bu metot içerisinde, işlevsel süreç keşfi için kullanılacak tetiklenme zamanı ve hareket türü (Giriş) de kaydedilmektedir.

NotifyEntryMovement() metodu arayüzden gelen tetiklemelerin ele alındığı metotlar içerisinden çağrılır. Normal akışa müdahale etmeden yalnızca ilgili kodun eklenmesi yeterli olacaktır. Tetikleyici olay parametre olarak geçilecektir. Örnek olarak "Giriş" butonuna tıklandığında tetiklenen olay neticesinde yapılacak işlemleri ele alan metot içerisinde kullanıma aşağıda yer verilmiştir.

```
private void GirisButtonActionPerformed(java.awt.event.ActionEvent evt) {
    ...
    m.NotifyEntryMovement(evt);
    ...
}
```

NotifyDBMovement(): GKU üzerinden gelen hareketlerin veri tabanına erişim yaptığı noktalarda sorgu cümleciği parametresi ile çağrılır. Öncelikle gerçekleşme zamanı kaydedilir ve bu zaman, son gelen Giriş hareketinin gerçekleşme zamanı ile karşılaştırılır. Geçen süre belirlenen sürenin altında ise işlevsel süreç oluşturabilecekleri varsayılır. Sonrasında sorgu cümlesi parçalanarak, sorgunun Okuma (“Read”) ve Yazma (“Write”) hareketlerinden hangisini gerçekleştirdiği anlaşılır ve kayda alınır.

NotifyDBMovement metodu veri tabanı üzerinde işlem yapacak noktalarda veri tabanı çağrısından önce eklenir. Veri hareketinin türünü belirleyebilmek için parametre olarak sorgu cümlesini alır. Örnek kullanıma aşağıda yer verilmiştir.

```
...
    query = "select * from Table1 where ornekID = '"+id+"'";
    m.NotifyDBMovement(query);
    res = st.executeQuery(query);
...
```

NotifyExitMovement(): Veri tabanından dönen değerlerin GKU üzerinde gerçekleştirdikleri hareketler, değişim dinleyicileri yardımıyla dinlenir. Herhangi bir değişim gerçekleştiğinde bu metot çağrılır. Gerçekleşme zamanına göre bir işlevsel sürece ait olup olmadığı belirlenir ve ona göre Çıkış (“Exit”) hareketi kayıt işlemi gerçekleştirilir.

NotifyExitMovement() metodu veritabanı işlemleri sonrasında arayüze dönen yanıtların tetiklediği metotlar içerisinden çağrılır. Normal akışa müdahale etmeden

```
public void propertyChange(PropertyChangeEvent evt) {
    m.NotifyExitMovement();
    ...
}
```

yalnızca ilgili kodun eklenmesi yeterli olacaktır. Bu metot parametre almaz. Örnek olarak veritabanından gelen bir cevap sonrası tetiklenen metot içerisindeki kullanım verilmiştir.

CalculateCosmic(): GKU'dan çıkış sırasında çağrılır. Uygulama çalışırken kayda alınan ve işlevsel süreç oluşturduğu doğrulanan hareketler kayıtlardan okunarak sayılır. COSMIC yaklaşımında belirlendiği şekilde, her bir hareket 1 cibb olarak hesaplanır. Sonuçlar kullanıcıya; GKU'nun sunduğu işlevsellik içerisinde keşfedilen işlevsel süreçler, bu süreçleri oluşturan veri hareketleri ve ölçülen yazılım büyüklüğü olarak rapor halinde sunulur.

CalculateCosmic() metodu programdan çıkış yapmadan hemen önce tetiklenecek şekilde yerleştirilmelidir. Bu konu uygulamadan uygulamaya değişiklik göstermekle birlikte en yaygın kullanım Frame üzerindeki  butonu kullanılarak yapılan kapatma işlemidir. Frame üzerinde önceden tanımlanmış WindowListener'ın windowClosing() metodu içerisinden CalculateCosmic() metodu çağrılır. Uygulama üzerinde çıkış için ayrıca bir buton olması durumunda, ilgili butona basılması sonrasında tetiklenen metot içerisinden çağrılabilir. Aşağıdaki örnekte Frame üzerinden kapatılma durumunda eklenen kodu göstermektedir.

```
public void windowClosing(WindowEvent e) {
   ...
   m.CalculateCosmic();
   ...
}
```

2- Kaydetme ve Değerlendirme Evresi

a) Veri Hareketlerinin Tetiklenmesi

"Veri Hareketlerinin Tetiklenmesi" evresi, işlevselliğe ait veri hareketlerinin kullanıcı tarafından, uygulama arayüzü üzerindeki (GKA) arayüz bileşenleri vasıtasıyla tetiklenerek bildirildiği evredir. Measurement Kütüphanesi kullanılarak ölçme işleminin gerçekleştirilebilmesi için ölçümü yapılacak tüm işlevselliğin kullanıcı

tarafından arayüz üzerinden tetiklenmesi gerekmektedir. Bu evre, sürecin manuel olarak gerçekleştirilen son adımıdır.

b) Veri Hareketlerinin Kaydedilmesi

Veri Hareketlerinin kaydedilmesi evresinde, bir önceki evrede yaptığımız kod eklemeleri sayesinde uygulama çalışırken gerçekleşen temel veri hareketlerinin izlenebilmesi sağlanmıştır. Measurement Kütüphanesi içerisinde bu veri hareketleri daha sonra değerlendirilebilecek şekilde kayıt edilmektedir. NotifyEntryMovement() metodu oluşan girdi veri hareketini gerçekleşme zamanı ile birlikte kayıt altına alır. NotifyDBMovement() ise öncelikle veri tabanına yapılan hareketleri sorgu cümleciğini parçalayarak belirler. Sonrasında hareketin oluş zamanı ile birlikte kaydeder. Benzer şekilde NotifyExitMovement() çıktı veri hareketlerinin izlenerek kaydedilmesini sağlar.

c) İşlevsel Süreç Keşfi

Veri hareketleri, hareketin türü (entry, exit, read, write) ve hareketin gerçekleşme zamanı bilgileriyle birlikte saklanmaktadır. Hareketlerin gerçekleşme zamanları kullanılarak işlevsel süreç oluşturup oluşturmadıkları belirlenmeye çalışılır. Measurement Kütüphanesi bir işlevsel sürecin 1 saniye içerisinde gerçekleşeceğini kabul etmiştir. (Bu değer uygulamadan uygulamaya ve sistemden sisteme değişebilen bir değer olmakla birlikte, kullanıcı tarafından parametrik olarak girilebilmesi ileriki çalışmalarda sağlanacaktır.) Aynı saniye içerisinde gerçekleşen hareketler aynı işlevsel sürecin bir hareketi olarak değerlendirilir. COSMIC metoduna göre bir işlevsel süreç biri Entry diğeri ise Read ya da Write olmak üzere iki veri hareketinden oluşur. Dolayısı ile Measurement Kütüphanesi ortada bir entry yok iken oluşabilen diğer hareketleri ihmal etmektedir.

3- Hesaplama Evresi

a) Veri Hareketlerinin Sayılması

İşlevsel süreçlere göre veri hareketleri kaydedildikten sonra uygulama sonlandırıldığı sırada ekran bazında gerçekleşen veri hareketleri sayılır ve her ekran için (Farklı Dialog, Frame ya da Tablar'daki) işlevsel büyüklük "Measurement Results.doc" dosyası altında raporlanır. Örnek bir uygulama sonrasında (Bankacılık Uygulaması) dosyada oluşan kayıtlara aşağıda yer verilmiştir.

```
Cosmic Value for screen "Giriş Ekranı" = 3
Cosmic Value for screen "Hesaplarım" = 6
Cosmic Value for screen "Güncel Faiz Oranları" = 3
Cosmic Value for screen "Para Transferleri" = 6
Cosmic Value for screen "Kredi Kartlarım" = 9


Total Cosmic Value : 27
```

SONUÇ

Measurement Kütüphanesi yazılımların işlevsel büyüklüğünün uygulama zamanında otomatik olarak ölçülmesini sağlayan bir araçtır. Ölçme sürecinin yazılım geliştirme sürecine entegre edilebilmesine olanak sağlayan bu araç, anlatılan kısıtlar dahilinde kullanılabilmektedir. Yazılım geliştirme sürecinin en başından itibaren ilgili kod eklemelerinin geliştirme ile birlikte devam ettirilmesi kütüphanenin kullanımından sağlanan başarımı en yüksek düzeye çıkaracaktır. Ölçmenin yapılacağı yazılımı belirlerken, kod eklemelerini yaparken ve sonuçları değerlendirirken bu kullanım kılavuzunda yer alan bilgiler iyi bir yol gösterici olacaktır.

Measurement Kütüphanesi geliştirilmesi için yapılacak katkılar yazılım mühendisliğinde ölçme alanında ilerleme kaydedilmesi noktasında büyük önem taşımaktadır. Kütüphanenin kullanımı konusundaki sorularınızı ya da gelişimi noktasındaki geri bildirimlerinizi measurementlibrary@gmail.com üzerinden iletebilirsiniz.