Hacettepe Üniversitesi Bilgisayar Mühendisliği Bölümü

Bil 445 Yazılım Mühendisliği Kavramları Dersi

5. Ödevi

Ad : Serdar
Soyad : Gül
No : 20421689
Mail : fuchserdar@gmail.com

Yazılım Ürünü Ölçümü

Yazılım ölçümü ; yazılımın miktarı , oranıdır. İlkel kullanımı ; yazılım geliştirmenin planlanması ve tahmin edilmesi üzerine idi. Eğer yazılımı nitelendirebilirsek , yazılımın kalitesini ve gelişim etkisini kontrol etme imkanına sahip olabiliriz. Bu özellikle yazılımın erken aşamaları için son derece doğrudur. Araştırma , yazılım ölçülerinden yazılım bakım etkisini tahmin etme alanında kullanılmaktadır.

Mesela Li ve Henry 1993'te 2 adet Klasik-Ada Sistemini kullanarak nesneye yönelik ölçülerin toplamından yola çıkarak başarılı bir şekilde bakım etkisini tahmin edebilmişlerdir.

Basili aynı ölçülerden yola çıkarak 1996 yılında öğrenci mezuniyet projelerini kullanarak tasarım kalitesini tahmin edebilmiştir.

Genel olarak yazılım ölçüleri , yazılım ürünü ölçüleri ve yazılım süreci ölçüleri olmak üzere 2'ye ayrılır. Yazılım ürünü ölçüleri ; kaynak kod ve tasarım belgeleri gibi yazılım ürününün miktarlarıdır. Yazılım süreci ölçüleri ise ; yazılım geliştirme sürecinin ölçümleri , miktarlarıdır.

Örnek olarak örneğin yazılımın boyutu , yazılım ürünü ile ilgili bir ölçüdür ve bu yüzden yazılım ürünü ölçüsüdür.Bir yazılım sistemi tasarlamak için gerekli olan çaba ise yazılımın nasıl geliştirildiğinden etkilenir. Bu sebeple yazılım süreci ölçüsüdür. Bu makalede sadece yazılım ürünü ölçüleri ile ilgili adresleme ve bilgi paylaşımı yapılacaktır.

Bir yazılım sistemi , diğer sistemler gibi , dahili bir tasarım yapısına ve harici davranışlara sahiptir. Bir kelime işlemcinin görünmesi ve hissedilmesi o yazılımın harici bir davranışıdır. Dahili olarak ise , yazılım ; bazı yazılım modüllerini bir araya getirerek inşa edilen ve tasarlanan bir üründür. Bu da yazılımın dahili tasarım yapısıdır.

Bir yazılımın dahili tasarım yapısı ; sistemi yerine getirmek için kullanılan programlama dilinin modeline bağlıdır. Yazılım sanayisinde işlevsel ve nesneye yönelik olmak üzere 2 çeşit programlama modellemesi yaygın olarak kullanılmaktadır.

Pascal ve C , işlevsel modellemeye örnek olurken , SmallTalk ve JAVA ise nesneye yönelik modellemeye örnek teşkil eder.

Yazılımın dahili yapısı bu iki modelleme içinde biraz farklılık gösterir. Mesela fonksiyonlar ve fonksiyonlar arası geçişler popüler bir C programının yapısını oluşturmak için kullanılırken ; SmallTalk veya JAVA , sistemlerini sınıflar , nesneler ve nesneler arası geçişlerle oluşturur.

Haliyle farklı modellemelerdeki yazılım ürünlerini ölçecek ölçümler de farklı olacaktır. İşlevsel modellemede , ölçümler fonksiyonları ve fonksiyonların nasıl etkileşim içinde olduğunu ölçer.Sınıflar ve iletişimlerinin ölçülmesi ise nesneye yönelik modellemede yapılan işlemlerdir. İşlevsel modellemede bir fonksiyonun imzası(isim,tür,parametre listesi) , fonksiyonların diğer fonksiyonlarla iletişime fonksiyon çağrılarıyla iletişime geçmesi için olabildiğince iyi olmalıdır , sistemin dahili yapısını oluşturur.

Nesneye yönelik modellemede , bir sınıfın verisi ve işlemlerin nitelikleri ve sınıfların diğerleri ile nasıl eşleştiği , yazılımın statik yapısını oluşturur. (Nesneye yönelik bir yazılım sistemi elbette nesneler ve birbirleri ile iletişimini de barındırır. Ve bu da yazılımın dinamik yapısını oluşturur. Nesneye yönelik sistemlerin dinamik modellemesi ise bu makalenin kapsamını aşan bir konudur.)

İşlevsel Modellemede Ölçüm

İşlevsel modellemedeki yazılım yapılarında 2 görüş öne çıkar. Biri her fonksiyon veya yordam içindeki kontrol akışıdır. Diğeri ise fonksiyonlar arası geçişlerdir. İşlevsel bir yapıyı ölçmek için kullanılan ölçümler ; fonksiyonun boyutu , bir fonksiyondaki kontrol akışı karmaşıklığı , bilgi akışı karmaşıklığıdır.

Fonksiyonun boyutu , tartışmalı ama çok kullanılan bir ölçüm birimidir. Tartışmalıdır , çünkü herkesin ortak onay verebileceği bir mükemmel boyut ölçümü yoktur. Buna mukabil çeşit çeşit ölçüm birimi bulunur , bunlardan biri de Kod Satırı (LOC)dır. Kod Satırı , bir fonksiyon içindeki bütün kaynak

satırları , boşluklarla ve yorum satırlarıyla veya onlarsız olarak hesaplanır. Boşluk ve komut satırlarının hesaplamaya dahil olup olmayacağı uygulayıcıya ve detaylı genel duruma bağlıdır. Genelde bu bir tanımı sürekli olarak kullandığımızdan dolayı pek de sorun olmaz.

Makalede verilen örnek C program kodunda hesaplamalarda boşluklar ve yorum satırları da hesaplamaya dahil edilmiştir. Ve hesaplana değerler makalede bir tabloda gösterilmiştir.

1976'da McCabe Cyclomatic Karmaşıklığı adında bir ölçü birimi (bir fonksiyon içindeki kontrol akışı karmaşıklığını ölçmek için) buldu. Bu ölçüm de bir fonksiyonun basitleştirilmiş akış çizelgesinde sınırlandırılmış alanları sayarak toplamına eklemek suretiyle hesaplanıyor.

Burada makaledeki şekilde ilgili bütün yerlere numaralar veriliyor ki hesaplamalar kolayca yapılsın. Bu şekle göre main fonksiyonu içinde 4 adet kesim olduğu için main() fonksiyonunun Cyclomatic karmaşıklığı (CC) 5 oluyor. Bir fonksiyonun CC ölçümü , 2 farklı şekilde ölçülebilir.

* Fonksiyon içindeki tek karşılaştırma belirtimlerinin sayısı
* CC = E – N + 2 (E : kenar sayısı , N : fonksiyonun basitleştirilmiş akış çizelgesindeki düğüm sayısı)

Bu ölçüm bize bir fonksiyonun kontrol akışının ne kadar karmaşık olduğunun ispatı olarak geri döner. Ayrıca bu fonksiyonun basit testi için ne kadar test elemanı gerektiğinin de ispatı olur bizim açımızdan. CC karmaşıklıkları da makalede aynı tabloda belirtiliyor.

Hem boyut hem de CC ölçümü bir fonksiyon içinde ölçülebilir. Buna rağmen boyut ölçümü bütün sistem için de genişletilebilir. Bu da sistemdeki bütün fonksiyonlar için Kod Satırı değerlerini ölçüp toplamak demektir.

Henry Kafura Bilgi Akışı Karmaşıklığı ölçümü , bir fonksiyonun giriş ve çıkış yelpazesine dayanan bir ölçümdür. Bu ölçüm bir fonksiyonun sistem içindeki çevresi ile ne kadar etkileşimde bulunduğunu hesaplayan bir ölçümdür.

Giriş yelpazesi , diğer bir fonksiyona direk çağrı , karar vermek için değeri kullanılacak bir değişken , veya direk olmayan bir yerel akış olabilir. Direk olmayan yerel akış , bir

fonksiyon başka bir fonksiyonu çağırdığından ve çağrılan fonksiyondan çağırma rutinine bir değer döndüğünde sonuçlanır. Çağırma rutini dönüş değerini ikinci bir fonksiyona gönderir. Bu da ilk çağrılan fonksiyon ile ikinci çağrılan fonksiyon arasındaki direk olmayan yerel akıştır.

Çıkış yelpazesi ise bir başka fonksiyona direk çağrı veya fonksiyonla ya da direk olmayan yerel akış ile değeri değişebilecek bir global değişken olabilir. Makaledeki örnekte de görüldüğü gibi ilk çağrılan fonksiyon çıkışında direk olmayan yerel bir akışa sahiptir ikinci çağrılan fonksiyonun içine gelecek şekilde. Bilgi akışı karmaşıklığı da giriş ve çıkış yelpazesinin toplamının 2 katı olarak hesaplanır. İlk tabloda bununla ilgili veriler de bulunmaktadır. Ve bu sayılar dahil ettiğimiz kütüphanelerin yazıldığı satırları da içermektedir.

Nesneye Yönelik Modellemede Ölçüm

Nesneye yönelik sistemler , işlevsel benzerlerine göre daha farklı bir yapıya sahiptir. Bu da 2 farklı modellemenin 2 farklı yapı taşına dayanmasından oluşur. Biri sınıflar, diğerleri de fonksiyonlardır. Bu sebeple nesneye yönelik ölçümler de farklı olmaktadır. Childamber ve Kemerer nesneye yönelik yapının ölçümü için 6 adet ölçü birimi geliştirmişlerdir. Bunlar ;

-Kalıtım Ağacının Derinliği (DIT)
-Çocuk Sayısı(NOC)
-Bir Sınıfın Ağırlıklı Metodu (WMC)
-Bir Sınıf İçin Cevap(RFC)
-Nesneler Arası Eşleştirme(CBO)
-Metotlardaki Birbirine Bağımlılık Eksikliği(LCOM)

Bu takım Li tarafından sonra tekrar gözden geçirilecek. Şimdilik biz sadece üzerinden geçeceğiz.Nesneye yönelik bir tasarımda bir sınıf , bir kalıtım hiyerarşisi içerisinde bulunur. Buradan Ata Sınıfların Sayısı (NAC) özelliği hesaplanır ki bu sayı , hiyerarşi içinde kaç ata sınıfı olduğu bilgisini de verir bize. Türeyen Sınıf Sayısı (NDC) adlı birim de bize o sınıftan kaç adet

sınıf türediğini yani torun sınıfları (aynı hiyerarşide) belirtir. Yerel Metot Sayısı (NLM) adlı birim , bize sınıfın yerel ara yüzünün ne kadar geniş olduğunu anlamamızı sağlar. Sınıf Metodu Karmaşıklığı (CMC) ise bir sınıfın içindeki bütün metotların birleşik karmaşıklığı bilgisini verir bize.

Soyut Veri Yapılarında Eşleşme (CTA) birimi , bize veri özelliği belirtiminde soyut veri yapısı kullanan sınıf sayısını verir. Mesaj İletimi ile Eşleşme (CTM) birimi ise bir sınıfın nesnesinin işletim sırasında sisteme kaç adet mesaj gönderdiği bilgisini bize verir. Makaledeki 2 nolu tablo burada bahsettiğimi değerleri yukarısında belirtilen örnek kod için hesaplamıştır.

NAC , bir sınıfın tasarımında kalıtım yüzünden kaç sınıfın bilgilerinin kullanılması gerektiğini hesaplamaya yarıyor. NDC de sınıfın etkileyebileceği torun sınıf sayısını hesaplıyor. Kalıtım , yeni bir sınıf tasarlarken daha önce üretilmiş sınıfların tekrar kullanılmasını sağlıyor. Ama bu aynı zamanda eşleşmeyi de sağladığından ata sınıfta olabilecek bir değişiklik , bütün torun sınıfları da doğal olarak etkiler. Bu yüzden NAC ve NDC kalıtım hiyerarşisinde olası çoğalabilecek değişimi hesaplamak için son derece önemlidir. Ve bu ölçümler sınıfın bir birimidir.

NLM bir sınıfın yerel ara yüzünün boyutunun ölçmek için tasarlanmıştır. Bir sınıf , kendi veri nitelikleri için bir sarmalamayı destekler. Sınıfta yerel olarak yer alacak public metotlar bize kalıtım etkisinden etkilenmemiş bir sınıf ara yüzü ölçümü hususunda yardımcı olacaktır. Bu sebeple NLM bir metot birimidir.

Bir sınıf diğer sınıflarla 3 farklı şekilde eşleşebilir.Bunlar
-Kalıtım
-Soyut veri yapıları
-Mesaj gönderme dir.

Ne zaman bir sınıf diğerini direk veya direk olmadan kalıtsa iki sınıf eşleşmiş olur. Ve bu da nesneye yönelik modellemenin sağladığı sarmalama olayını kırar. NAC ve NDC burada etkilidir.

İki sınıf eğer biri diğerini soyut bir veri yapısı olarak kullanırsa da eşleşmiş olur. Makalede verilen örnekte A sınıfı ile B sınıfı arasındaki ilişki buna bir örnektir(figür 2). CTA birimi de bu durumlarda işe yarar.

İki sınıf , eğer bir sınıfın nesnesi , diğer sınıfın nesnesine mesaj (ileti) gönderirse de bu iki sınıf eşleşmiş olur. Kalıtım ve soyut bir veri yapısı kullanımı zorunlu değildir. Makalede verilen örnekte figür 3te A sınıfı ile B sınıfının arasındaki ilişki bu türdendir.A sınıfının nesnesi B sınıfının nesnesine mesaj göndermektedir.

Bir sınıf için mesaj göndererek iletişim kurma açısından 2 görüş vardır. Biri sınıfın diğer sınıflardan alabileceği mesaj sayısının derecelendirilmesi , diğeri de bir sınıfın diğer sınıflara yollayabileceği mesajların kapsamıdır. NLM ve NAC'ın bazı kombinasyonları sınıfın alabileceği mesajlarım kapsamını hesaplamakta kullanılabilir. Buna rağmen mesaj göndererek iletişim tartışması sınıfın diğer sınıflara göndereceği mesaj kapsamı ile sınırlandırılmıştır. CTM de bu tip sınıflar için kullanılır.

Özet

Yazılım ölçümleri göreli olarak genç bir bilim ve mühendislik disiplinidir. Diğer kurulmuş disiplinler gibi bunun da oturması için bir miktar sürenin geçmesi gerekir. Bu ölçümler bu alanın esaslarını kavramak adına yeterince irdelenmiştir. Görev yüzlü yazılım ölçümleri araştırması da bu ölçümlerin doğrulamasını oluşturur. Eğer bu ölçümleri endüstriyel yazılım geliştirmede etkili olarak kullanmak istiyorsan bunlar anlamlı ve kullanışlı olmalıdır. Yazılımlar için bu tip bazı ölçümleri yapmak hiç de zor değildir. Zor olan kısım geçerleme ve doğrulamayı yapmaktır.

Teşekkürler

Dr. Letha Etzkorn , Bay Amirali Jalia ve Bay Mohammed Alshayeb'e yardımlarından ötürü teşekkür ederim.

Yazar Hakkında

Wei Li Virginia Üniversitesinden Bilgisayar Bilimleri dalında Profesör Doktor ünvanını aldı. Şu anda Huntsville'de Alabama Üniversitesinde asistan profesör olarak çalışmaktadır. İlgi alanları nesneye yönelik programlama , yazılım ölçümleri , tekrar kullanılabilir bileşenlerdir. Dr. Li şu anda ACM ve IEEE Bilgisayar Topluluğunun da bir üyesidir.