Çok Katmanlı Veritabanı Uygulamaları İçin Esnek Bir Vb.Net Kodu Üreticisi: "Code Generator"

[1]Mustafa YILDIZ[*], [2]Orhan KARAHASAN, [3]Selahattin KURU

[1]Teknopazar A.Ş., ITU Ayazağa Kampüsü, ARI Teknokent No:9, 34469 Maslak, İstanbul

[2,3]Işık Üniversitesi, Enformatik Uygulama ve Araştırma Merkezi
Büyükdere Caddesi, 34398 Maslak, İstanbul

[1]mustafa.yildiz@is.net.tr, [2]orhan@isikun.edu.tr, [3]kuru@isikun.edu.tr

Özet

Birçok yazılım mühendisliği uzmanının kaynak kodun okunabilirliğini, yönetilebilirliğini ve tekrar kullanılabilirliğini belirgin biçimde artırdığı savı ile savunduğu, desteklediği ve önerdiği çok katmanlı yazılım mimarisi yakın zamanda Microsoft firması tarafından da biçimsel bir mimari önerisi ile desteklenmiştir. Bu mimariye göre yazılım, kullanıcı arayüzleri seviyesinden itibaren veritabanı seviyesine kadar çok sayıda mantıksal katmana ayrılmış ve saklanmış yordam (stored procedure) kullanımı teşvik edilmiştir. Bu çalışmada, önerilen bu çok katmanlı mimari doğrultusunda veritabanı katmanı, veri erişim katmanı, iş varlıkları katmanı ve iş mantığı katmanı olmak üzere dört farklı katmana ait kaynak kodlarını büyük ölçüde otomatik olarak üreten bir kod üreticisi geliştirilmiştir. Bu bildiride önerilen mimarinin detaylarına, geliştirilen kod üreticisinin özelliklerine ve kod üreticisinin verimini ortaya koyan bazı ölçüm sonuçlarına değinilmektedir.

Abstract

Many software engineering experts supports and recommends the use of n-tier architecture due to its significant contributions to the readability, maintainability and reusability of the source code. Recently Microsoft declared a formal architecture description on the use of n-tier architecture clearly dividing the code into logical layers from presentation tier to data access tier and encouraging the use of stored procedures. In this work a code generator tool was developed which produces code for database layer, data access layer, business entities layer and business logic layer. This paper reports the details of the tool and results of some measurements regarding the efficiency of the tool.

1. Giriş

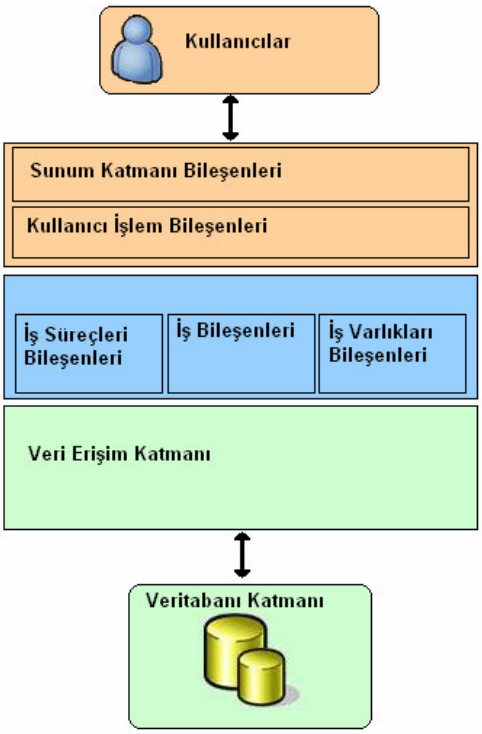
Bu bildiri çok katmanlı mimariye uygun olarak geliştirilen veritabanı uygulamalarının veritabanı katmanı, veri erişim katmanı, iş varlıkları katmanı ve iş mantığı katmanına ait kaynak kodları belirli bir oranda otomatik olarak üreten esnek bir kod üreticisini tanıtmaktadır.[1] Geliştirilen kod üreticisi araç, Microsoft firmasının önerdiği çok katmanlı mimari standartlarına uygun olarak çalışmaktadır ve orta büyüklükte bir veritabanı uygulaması yeniden mühendisliği projesinde denenmiştir. Proje tek katmanlı olarak geliştirilmiş olan internet tabanlı bir finans uygulamasının çok katmanlı olarak yeniden yazılmasını içermektedir.

Bildiride Microsoft tarafından önerilen çok katmanlı mimari anlatılacak ardından kod üretici aracın işlevleri kod ürettiği katmanlar üzerinden açıklanacaktır. Aracın verimi çeşitli katmanlardaki toplam kod uzunluğuna karşın otomatik olarak üretilen kod miktarları verileri ile desteklenmiştir. Bu

[*] Çalışma Işık Üniversitesi, Enformatik Uygulama ve Araştırma Merkezi'nde yapılmış olup Mustafa Yıldız çalışma sırasında bu merkeze bağlı olarak çalışmaktaydı.

veriled detaylı şekilde sonuç bölümünde verilmiştir. Geliştirilmiş olan araç internet üzerinden erişilebilir durumdadır ve açık kaynak kodlu olarak dağıtılması planlanmaktadır.

2. Microsoft Tarafından Önerilen Çok Katmanlı Mimari



Bu bölümde anlatılacak olan mimari Microsoft'un önerdiği ve örnek uygulamalarla açıkladığı çok katmanlı mimari olup, bu mimarinin esasları ve faydaları üzerinde durulacaktır.[2] Microsoft'un önerdiği uygulama mimarisinde yandaki şekilde de görüleceği üzere yedi farklı bileşen yer almaktadır. Bunlar sırasıyla aşağıda verilmiştir.

Sunum Katmanı Bileşenleri: Hemen hemen tüm yazılım ürünleri kullanıcılara bazı bilgileri vermek, bazı bilgileri almak üzere arayüzler sağlamak durumundadırlar. Örneğin bir e-ticaret sisteminde, müşteriler ürünleri görebilmekte, değişik filtrelere göre arama yapabilmekte yada seçtiği ürünlerden sipariş oluşturup müşteri temsilcilerine gönderebilmektedirler. Sunum katmanı bileşenleri genellikle web sayfaları, windows formları şeklinde olmakta ve kullanıcıya bazı bilgiler vermekte, yada doğruluğunu kontrol ettikleri bilgileri diğer katmanlara ulaştırmaktadırlar.

Kullanıcı İşlem Bileşenleri: Çoğu kez kullanıcıların takip edecekleri işlem sırası önceden belirlenmiştir. Örneğin, e-ticaret uygulamasında ürünler belirli kategorilere ayrılmış ve belli bir ürüne erişebilmek için bir kategori seçilmesi ardından ürünün seçilmesi gerekiyor olabilir. Benzer şekilde müşteri bir satınalma yapmak istediğinde takip etmesi gereken işlemler bellidir. Öncelikle ürün bilgileri girilir ardından ödeme bilgileri ve en son da adres bilgileri şeklinde olabilir. Farklı kullanıcı işlem bileşenleri kullanarak kullanıcı etkileşimlerinin senkronizasyonuna yardımcı olunabilir. Bu sayade işlem akış ve durum yönetimi daha kolay olacak ve işlem bileşenleri birden fazla kullanıcı arayüzü tarafından kullanılabilecektir.

İş Süreçleri Bileşenleri: Kullanıcı işlem bileşenleri tarafından bilgi toplandıktan sonra, veri iş süreçlerinde kullanılmaktadır. Bu katman bileşenleri, belirli bir düzene göre takip edilmesi gereken işlemleri yapmak için kullanılırlar. Örneğin, e-ticaret sisteminde gerekli bilgiler alındıktan sonra system toplam sipariş miktarını hesaplamak, kredi kartı bilgilerini doğrulamak, kredi kartından para çekme işlemlerini gerçekleştirmek, ve ürünlerin gönderilmesini planlamak durumundadır. Tüm bu işlerin takibi İş süreçleri Bileşenleri tarafından yapılır.

İş Bileşenleri: Bir işin birden fazla basamakta yapılmasına bakılmaksızın, uygulamalar iş kurallarını uygulayan ve iş görevlerini yerine getiren bileşenlere ihtiyaç duyarlar. Örneğin e-ticaret uygulamasında, bir siparişin değerini hesaplayan bir bileşene ihtiyaç vardır.

Veri Erişim Katmanı: Çoğu uygulamalar ve servisler iş süreçlerinde belirli bir yerde duran veriye erişmek ihtiyacındadırlar. Örneğin, bir e-ticaret uygulamasında, ürün bilgilerine erişmek ve bu bilgileri müşterilere iletmek durumundadırlar. Veri erişim katmanının ayrı olması bakım ve

configürasyon kolaylığı getirmektedir. Ayrıca, veritabanı yönetim sisteminin değişmesi bu sayede diğer katmanlarda değişiklik olmadan kolaylıkla veri erişim katmanının değiştirilmesi yoluyla uygulama kolaylıkla değişikliğe adapte edilebilmektedir.

İş Varlıkları Bileşenleri: Çoğu uygulamalar bileşenler arasında veri iletişimi ihtiyacı hissederler. Örneğin, e-ticaret uygulamasında ürün listesi veri erişim bileşenlerinden sunum katmanı bileşenlerine gönderilmektedir. Genellikle gerçek dünyadan bazı varlıkları temsil etmektedirler (örneğin sipariş, ürün gibi). Uygulamalarda kullanılan iş varlıkları genellikle veri yapıları; verikümeleri(dataset), veri okuyucuları (datareaders) şeklinde olabileceği gibi gerçek hayatta varlıkları temsil eden sınıflar da olabilmektedir.

Güvenlik, İletişim ve Operasyonel Yönetim Bileşenleri: Uygulamalar genellikle istisna kotarım yönetimi(exception handling management) bileşenleri, kullanıcı yetkilendirmesi ve diğer servisler yada uygulamalarla iletişim bileşenleri kullanabilmektedir. Bunlara örnek olarak geliştirilmiş bir şifreleme sisteminin uygulamada kullanılması gösterilebilir.
Örnek uygulama geliştirilirken bu katmanlardan bazılarına ihtiyaç duyulmamıştır. İş süreçleri bileşeni bunlardan birisidir. Uygulamada takip edilmesi gereken, belirli bir düzende bir çok işlemin ardarda geldiği bir hal sözkonusu olmadığı için kod uygulama bu katmana ait kod içermemektedir.

3. Kod Üreticisi

Kod üreticisi veritabanı katmanı, veri erişim katmanı, iş varlıkları katmanı ve iş mantığı katmanı olmak üzere dört katman için kaynak kodlarının büyük bölümünü üretmektedir. Bu bölümde, kod üreticisinin işlevleri kodunu ürettiği katmanların herbiri için ayrı ayrı anlatılacaktır.
Veritabanı Katmanı

Önceki bölümde de değinildiği üzere, veritabanı katmanı, veritabanı tablolarındaki veriyi çeşitli şekillerde sorgulayan, yeni kayıtlar ekleyen, varolan kayıtları güncelleyen ve kayıtları silen saklanmış yordamları bünyesinde barındırır. Bu saklanmış yordamların büyük bölümü belirli bir şablona uyan ve kendisi ile ilgili tablonun sütunlarına ve bu sütunların tipine göre değişen bir yapı arzeder. *INSERT, UPDATE, DELETE ve SELECT* işlevlerini gerçekleştiren yordamlara ait şablonlar şu şekildedir.

```
--INSERT YORDAMI
CREATE PROCEDURE sproc_INSERT_<<tablo ismi>>
@<<sütun adı>> <<sütun tipi>>, [@<<sütun adı>> <<sütun tipi>>,]
@InsertedRecordID int output
AS
INSERT INTO <<tablo ismi>>
(<<sütun adı>>, [<<sütun adı>>,])
VALUES
(@<<sütun adı>>, [@<<sütun adı>>,])
Select @InsertedRecordID=@@IDENTITY
```

```
--UPDATE YORDAMI
CREATE PROCEDURE sproc_UPDATE_<<tablo ismi>>
@<<sütun adı>> <<sütun tipi>>, [@<<sütun adı>> <<sütun tipi>>,]
@RecordID
AS
UPDATE <<tablo ismi>>
SET <<sütun adı>> = @<<sütun adı>>, [<<sütun adı>> = @<<sütun adı>>,]
WHERE
<<anahtar sütun adı>> = @RecordID
```

```
--DELETE YORDAMI
CREATE PROCEDURE sproc_DELETE_<<tablo ismi>>
@RecordID
AS
DELETE FROM <<tablo ismi>>
WHERE
<<anahtar sütun adı>> = @RecordID
```

```
--SELECT YORDAMI
CREATE PROCEDURE sproc_SELECT_<<tablo ismi>>
@RecordID
AS
SELECT * FROM <<tablo ismi>>
WHERE
<<anahtar sütun adı>> = @RecordID
```

Yukarıdaki şablonların da açık bir biçimde ortaya koyduğu üzere veritabanı katmanında bulunan saklanmış yordamlar ilgili oldukları tabloların sütun isimleri ve sütun tiplerine bağlı olarak farklılık göstermekle beraber ortak bir şablonu paylaşmaları münasebetiyle otomatik olarak üretilmeye elverişlidirler. Tabloların sütun isimleri ve sütun tipleri *Microsoft SQL Server 2000* veritabanı yönetim sisteminde bulunan sistem tabloları sorgulanarak elde edilebilir. Bu bilgi diğer birçok veritabanı yönetim sisteminde de erişilebilir bir şekilde saklanmaktadır.

Veri Erişim Katmanı

Veri erişim katmanı, veritabanı katmanında bulunan saklanmış yordamları kullanan sınıflar ile bu sınıflara ait yordamlardan oluşur. Veritabanı katmanına benzer şekilde bu sınıf ve yordamlar ilişkili oldukları tabloların sütun isimleri ve tiplerine bağlı olarak farklılık gösterse de ortak bir yapı ve şablona sahiptir. Bu ortak şablon yine *INSERT, DELETE, UPDATE* ve *SELECT* yordamları için aşağıda ifade edilmiştir.

```
'Bir iş varlığı nesnesi kabul eden ve tabloya ekleyen yordamdır.
'Sonuç olarak eklediği kaydın anahtar değerini döndürür
Public Function Insert<<tablo adı>>(ByVal Obj<<tablo adı>> As <<tablo adı>>Info)
As Integer
<<Veritabanı Erişim Komutları>>
mycommand.CommandText = "sproc_INSERT_<<tablo adı>>"
mycommand.Parameters.Add("@<<sütun adı>>",<<sütun adı>>) .Direction =
ParameterDirection.Input
[mycommand.Parameters.Add("@<<sütun adı>>",<<sütun adı>>) .Direction =
ParameterDirection.Input]
mycommand.Parameters.Add("@InsertedRecordID", 1).Direction =
ParameterDirection.Output
mycommand.ExecuteNonQuery()
Return CInt(mycommand.Parameters("@InsertedRecordID").Value)
End Function
```

```
'Bir iş varlığı nesnesi kabul eden ve tablodaki kaydı güncelleye yordamdır.
'Sonuç olarak doğru/yanlış değeri döndürür
Public Function Update<<tablo adı>>(ByVal Obj<<tablo adı>> As <<tablo adı>>Info)
As Boolean
<<Veritabanı Erişim Komutları>>
mycommand.CommandText = "sproc_UPDATE_<<tablo adı>>"
mycommand.Parameters.Add("@<<sütun adı>>",<<sütun adı>>) .Direction =
ParameterDirection.Input
```

```
[mycommand.Parameters.Add("@<<sütun adı>>",<<sütun adı>>) .Direction =
ParameterDirection.Input]
mycommand.ExecuteNonQuery()
End Function
```

```
'Bir iş varlığı nesnesi kabul eden ve tablodaki kaydı silen yordamdır.
'Sonuç olarak doğru/yanlış değeri döndürür
Public Function Delete<<tablo adı>>(ByVal Obj<<tablo adı>> As <<tablo adı>>Info)
As Boolean
<<Veritabanı Erişim Komutları>>
mycommand.CommandText = "sproc_DELETE_<<tablo adı>>"
mycommand.Parameters.Add("@<<anahtar sütun>>",<<anahtar sütun>>) .Direction =
ParameterDirection.Input
mycommand.ExecuteNonQuery()
End Function
```

```
'Tablonun anahtar sütunu için tamsayı bir değer kabul eden ve tablodaki
'bu kaydın değerlerini bir iş nesnesine yükleyen yordamdır.
'Sonuç olarak yüklü iş nesnesini döndürür
Public Function Select<<tablo adı>>(ByVal RecordID As Integer) As Obj<<tablo
adı>>
<<Veritabanı Erişim Komutları>>
mycommand.CommandText = "sproc_SELECT_<<tablo adı>>"
mycommand.Parameters.Add("@<<anahtar sütun>>",<<anahtar sütun>>) .Direction =
ParameterDirection.Input
myreader = mycommand.ExecuteNonQuery()
If myreader.Read Then
If not isDBNull(myreader("<<sütun adı>>")) Then
     Obj<<tablo adı>>.<<sütun adı>> = myreader("<<sütun adı>>")
Else
     Obj<<tablo adı>>.<<sütun adı>> = 0
End If
End If
End Function
```

Veritabanı katmanına benzer şekilde veri erişim katmanının kaynak kodları belirli bir şablona uygun olmaları nedeniyle otomatik olarak üretilmeye elverişlidir. Bu katmanda üretilen kodlar bir alt katman olan veritabanı katmanının da bu araç ile otomatik olarak üretildiğini, bir programcı tarafından üretildi ise de en azından üreticinin standartlarına uygun olarak üretildiğini kati olarak kabul eder.

İş Varlıkları Katmanı

Önceki bölümde de detaylı olarak anlatıldığı üzere bu katmanda iş varlıklarının herbiri için bir sınıf bulunmaktadır. İş varlıkları veritabanı tasarımında veritabanı tabloları olarak büyük ölçüde ifade edildiğinden bu katmandaki sınıfların büyük kısmı veritabanı tablolarıyla birebir ilişkilidir ve bu tür sınıfların tümü ortak bir şablona sahiptir. İş varlığı sınıflarının ortak şablonu aşağıda ifade edilmiştir.

```
Public Class <<tablo adı>>Info
#Region "class variables"
    Private _<<sütun adı>> As ConvertType(<<sütun tipi>>)
    [Private _<<sütun adı>> As ConvertType(<<sütun tipi>>)]
#End Region

#Region "properties"
```

```
    Public Property <<sütun adı>>() As ConvertType(<<sütun tipi>>)
    Get
        Return _<<sütun adı>>
    End Get
    Set(ByVal Value As ConvertType(<<sütun tipi>>))
        _<<sütun adı>> = Value
    End Set
    End Property
#End Region
End Class
```

Dikkat edilirse bu sınıf, veritabanı tablosunda bulunan sütunların herbiri için özel (*private*) bir sınıf değişkeni ve bu değişkene erişim imkanı veren özellik yordamlarını (*properties*) içerir. Veritabanı sütunlarının tipleri (int, varchar, char, datetime, vb.) ile VB kodlarındaki değişken tiplerinin isimleri birebir aynı olmadığından *ConvertType* isimli bir tip ismi eşleştirme işlevi kullanılmıştır.

İş Mantığı Katmanı

Bu katmanda iş mantığını ortaya koyan çok çeşitli sınıflar bulunmaktadır. Bu sınıfların çeşitliliği bu katmanda geliştirilecek olan sınıfların belirli oranda programcılar tarafından kodlanmasını gerektirse de her sınıfta bulunması gereken ve bir alt katman olan veri erişim katmanındaki yordamlar ile iletişen temel yordamlar yine ortak bir şablonu paylaşır ve kod üreticisi yardımıyla otomatik olarak üretilebilir. Bu yordamlar yine temel *SELECT*, *INSERT*, *UPDATE* ve *DELETE* işlevlerine ait yordamlardır.

```
Public Shared Function Update<<tablo ismi>>(ByVal Obj<<tablo ismi>> As <<tablo
ismi>>Info) As Boolean
Dim Obj<<tablo ismi>>DAL As New <<tablo ismi>>DAL
Update<<tablo ismi>> = Obj<<tablo ismi>>DAL.Update<<tablo ismi>>(Obj<<tablo
ismi>>)
Obj<<tablo ismi>>DAL = Nothing
End Function
```

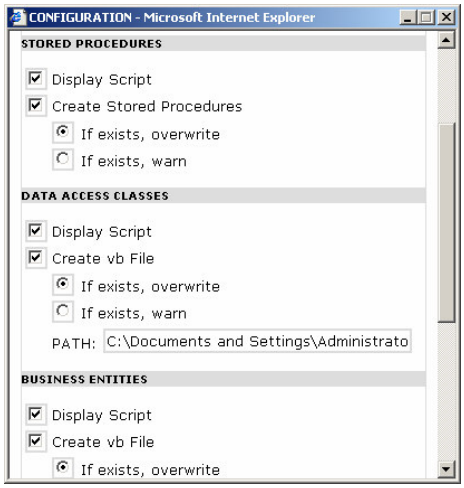
```
Public Shared Function Delete<<tablo ismi>>(ByVal Obj<<tablo ismi>> As <<tablo
ismi>>Info) As Boolean
Dim Obj<<tablo ismi>>DAL As New <<tablo ismi>>DAL
Delete<<tablo ismi>> = Obj<<tablo ismi>>DAL.Delete<<tablo ismi>>(Obj<<tablo
ismi>>)
Obj<<tablo ismi>>DAL = Nothing
End Function
```

```
Public Shared Function Select<<tablo ismi>>(ByVal RecordID As Integer) As
Obj<<tablo ismi>>Info
Dim Obj<<tablo ismi>>DAL As New <<tablo ismi>>DAL
Select<<tablo ismi>> = Obj<<tablo ismi>>DAL.Select<<tablo ismi>>(RecordID)
Obj<<tablo ismi>>DAL = Nothing
End Function
```

```
Public Shared Function Insert<<tablo ismi>>(ByVal Obj<<tablo ismi>> As <<tablo
ismi>>Info) As Integer
Dim Obj<<tablo ismi>>DAL As New <<tablo ismi>>DAL
Insert<<tablo ismi>> = Obj<<tablo ismi>>DAL.Insert<<tablo ismi>>(Obj<<tablo
ismi>>)
Obj<<tablo ismi>>DAL = Nothing
End Function
```

Şablonlarda da açıkça görüldüğü gibi bu katmandaki sınıflarda bulunması gereken yordamlardan dördünün kodu otomatik olarak üretilmeye elverişlidir.

İlave İşlevler



Kod üreticisinin önemli özelliklerinden biri de aracın oldukça esnek olması ve kullanıcıların birtakım ayarlamaları aracın arayüzlerinden kolaylıkla yapabilmesidir. Kullanıcı, üretilecek kodun temsili ile ilgili ayarlamalar yapabilmektedir.

Saklanmış yordamlara veya diğer katmanlardaki sınıf ve yordamlara ilişkin üretilmiş olan kodun veritabanına veya bilgisayarın diskine yazılması sırasında izlenecek olan kurallar kullanıcı tarafından belirlenebilmektedir.

Kullanıcı bağlanılacak veritabanına ilişkin erişim bilgilerini de yine aracın arayüzlerinden belirleyebilmektedir. Araç, şu an için yalnızca *Microsoft SQL Server 2000* veritabanı yönetim sistemi ile çalışmaktadır.

4. Sonuç

Bu çalışmada çok katmanlı mimariye uygun olarak geliştirilen veritabanı uygulamaları için kullanılabilecek esnek bir VB.NET kodu üreticisi olan "*Code Generator*" aracı geliştirilmiştir. Geliştirilen araç bir finans uygulamasının yeniden mühendisliğini içeren bir projede denenmiştir.

Yeniden mühendisliği yapılan finans uygulaması, bir firmanın Hazine işlemlerini gerçekleştirdiği internet tabanlı ve orta büyüklükte bir veritabanı uygulamasıdır. Yeniden mühendislik tek katmanlı olarak VB.NET ile geliştirilmiş olan bu uygulamanın çok katmanlı mimari kullanılarak yeniden geliştirilmesini içermektedir. Uygulamanın gereksinimleri halihazırda geliştirilmiş olan yazılımın kullanıcı arayüzleri ile tamamıyla ortaya konmuştur. Proje kapsamında geliştirilen uygulamaya ait kaynak kodu uzunlukları aşağıdaki tabloda katmanlara ayrılmış olarak verilmiştir.

KATMAN	KOD UZUNLUĞU (bin satır)
Sunum Katmanı	35
İş Mantığı Katmanı	18
İş Varlıkları Katmanı	18
Veri Erişim Katmanı	28
Veritabanı Katmanı	14
TOPLAM	**113**

Aşağıdaki tabloda ise her bir katmandaki kodun kod üreticisi ile otomatik olarak üretilen miktarı ile bu miktarı toplam kod uzunluğuna oranı verilmiştir.

KATMAN	OTOMATİK ÜRETİLEN KOD UZUNLUĞU (bin satır)	ÜRETİLEN KODUN TOPLAM KODA ORANI
Sunum Katmanı	0	% 0
İş Mantığı Katmanı	6	% 33
İş Varlıkları Katmanı	18	% 100
Veri Erişim Katmanı	24	% 86
Veritabanı Katmanı	13	% 93
TOPLAM	**61**	**% 54**

Yukarıdaki göstergelerde de görüldüğü gibi kod üreticisi geliştirilmiş olan toplam kodun 61 bin satırını yani yarısından fazlasını otomatik olarak üretmiştir. 61 bin satır kodun kabaca 8 adam/hafta'lık bir programcı emeğine denk geldiği düşünülürse aracın kullanılmasının projedeki verime katkısı daha net bir biçimde ortaya çıkmaktadır.

Kod üreticisine http://irdc.isikun.edu.tr/Code_Generator adresinden erişilebilmektedir. Aracın açık kaynak kodlu olarak paylaşılması da düşünelmektedir.

5. Referanslar

[1] Code Generator, http://irdc.isikun.edu.tr/Code_Generator

[2] "Application Architecture for .NET: Designing Applications and Services", http:// msdn.microsoft.com/library/en-us/dnbda/html/distapp.asp