COSEML'de İşbirliği Diyagramlarının Kullanımı

Mehmet Burhan TUNCEL

Bilgisayar Mühendisliği Bölümü, Fen Bilimleri Fakültesi, ODTÜ, Ankara

e-posta: mbtuncel@sbd.havelsan.com.tr

Özet

Yazılımın geleceği olarak görülen bileşen yönelimli yazılım mühendisliğinde (BYYM) modelleme ihtiyacı, mevcut modelleme standartlarıyla tam olarak karşılanamamaktadır. Bu ihtiyaca cevap vermek amacıyla yapılan çalışmalardan birisi olan COSEML (Component Oriented Software Engineering Modeling Language), nesneye yönelik modelleme yaklaşımlarının aksine, BYYM paradigması göz önüne alınarak tasarlanmıştır. Yaptığımız çalışmanın amacı geliştirme aşamasında olan COSEML'e işbirliği diyagramlarının dahil edilmesidir. Bu eklemeyle, bileşenler arasındaki etkileşimin detaylı olarak gösterilmesini ve modelden çalışır kod üretilebilmesini amaçlamaktayız. Çalışmanın ilk safhasında COSEML ile ilgili genel bilgi verilmiş ve sistemin nasıl modellendiği anlatılmıştır. Ardından işbirliği diyagramlarının modelleme ve gerçekleştirim aşamalarına sağlayacağı katkılara değinilerek sanal bir senaryo üzerinde örnek kullanımı gösterilmiştir.

1. Giriş

Sistemlerin her geçen gün daha büyük ve karmaşık hale gelmesiyle birlikte, yazılım mühendisliğinde modellemenin önemi de giderek artmaktadır. Akademik çevrelerde ve yazılım endüstrisinde kabul görerek standart haline gelen UML [1], nesneye yönelik yazılım geliştirme sürecine önemli katkılar sağlamıştır. Fakat UML, temel olarak farklı direkler üzerine kurulan BYYM gereksinimlerini yeterince karşılayamamaktadır. BYYM de amaç sistemi kod yazarak değil, mevcut bileşenleri bir araya getirerek oluşturmaktır [2]. Bileşen yönelimli geliştirmede, yukarıdan aşağıya ve aşağıdan yukarıya yaklaşımların birbirlerini destekler şekilde bir arada kullanılması önerilmektedir [3]. Yukarıdan aşağıya yaklaşımla sistem alt modüllere ayrılır. Aşağıdan yukarıya yaklaşımla ise mevcut bileşenler bir araya getirilerek alt modüllerin gereksinimlerini karşılayabilecek bileşenler meydana getirilir.

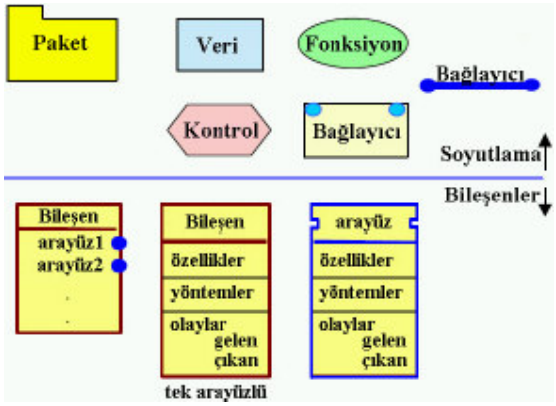
BYYM modellemelerindeki bu tür ihtiyaçları karşılayabilmek amacıyla COSEML geliştirilmiştir [4]. Bileşen yönelimli anlayışa uygun olarak tasarlanan bu modelleme dilinin gelişimi, yapılan akademik çalışmalarla devam etmektedir.

Bileşenler arasındaki etkileşimin ve iş mantığının daha detaylı modellenebilmesine olanak sağlayacak olan işbirliği diyagramlarının kullanımı, COSEML'e önemli katkı sağlayacaktır. Model içindeki yapısal ve bileşen seviyesindeki etkileşimin sırasal gösterimi hem modelin tanımlayıcılığını arttıracaktır, hem de çalışır kod üretilebilmesine olanak sağlayacaktır. Bu amaçla

sunduğumuz çalışma UML'de kullanılan collaboration (işbirliği) diyagramlarının [1], BYYM mantığına uyarlanarak ve çalışır kod üretebilme yeteneği de eklenerek COSEML'de kullanılmasıdır. Bu diyagramlarda uygulamayı planladığımız sırasal yöntem çağırmaya dayalı yapı, modelden kod üretimine oldukça uygun bir zemin hazırlamaktadır.

2. COSEML Hakkında

BYYM paradigması kod yazma yerine birleştirme yöntemiyle üretimi amaçlar. Henüz olgunlaşan bileşen teknolojisinden tam anlamıyla yararlanılamamakta, varolan yazılım mühendisliği yaklaşımları yetersiz kalmaktadır. Bu açığı kapatmak amacıyla, grafiksel bir modelleme dili olan COSEML geliştirilmiştir. Bu dilin geliştirilmesi amacıyla nesneye yönelik ve bileşen tabanlı modellemeler incelenmiş ve uygun olduğu yerlerde bu modellerde bulunan semboller uyarlanarak kullanılmıştır.
Şekil 1'de modelleme dilinde kullanılan temel semboller gösterilmiştir.



Şekil 1. COSEML de kullanılan semboller

Bileşen yönelimli tasarımda yukarıdan-aşağıya ve aşağıdan-yukarıya yaklaşımların bir arada kullanılarak uzlaştırılması problem teşkil etmektedir; bu problemi çözmek için COSEML dilinde hem soyut hem de bileşen seviyesinde semboller kullanılmıştır. Bu sayede, model içerisinde hem soyut tanımlamalar, hem de karşılık gelen bileşenler bulunabildiğinden bir sistemsel işlev farklı seviyelerde gösterilebilir. COSEML ile model oluşturma işlemi, sistemin yapıtaşlarını bulmak amacıyla yukarıdan-aşağıya başlar. Bu yapısal ayrıştırma işlemi, mevcut bileşenlere ulaşılıncaya kadar devam eder. Bu aşamadan sonra bileşenler, aşağıdan yukarıya bir yaklaşım izlenerek bir araya getirilir [4,5]. Böylece sistemin yapısal görüntüsü ve bu sistemi oluşturacak bileşenler aynı model üzerinde incelenebilmekte ve sistemin işlevleri farklı seviyelerde gösterilebilmektedir.

3. COSEML İle Modelle

Modelleme yapılmadan önce, sistemin uygulanacağı alana ait proje gereksinimlerine uyan hazır bileşenlerden oluşan bir bileşen havuzu oluşturulur. Daha sonra bu havuzdaki bileşenler de göz önüne alınarak, sistem yapısal ayrıştırma döngüsüne girer. Bu süreç havuzdaki bileşenlere ulaşılıncaya kadar devam eder. Sistemin yapısal ayrıştırmayla bileşen seviyesine ulaşması, doyuma

ulaşmış (standartları tanımlanmış ve alana ait tüm gereksinimleri karşılayan bileşen havuzuna ulaşılmış) bir alanda kolay bir süreç olacaktır.

Soyut modellemenin ardından, bileşen seviyesine geçilmeden önce, bu çalışmada önerdiğimiz işbirliği diyagramları kullanılarak ayrıştırılan sistem elemanları arasındaki etkileşim modellenebilir. Bu diyagramların kullanımı sistemin işleyişi ve istenilen işlevselliğin ne kadar karşılandığı gibi konularda bilgilendirici olacaktır [1]. Böylece uygun görülmeyen durumların oluşması modelin üst seviyelerinde fark edilerek modellemenin gözden geçirilmesine olanak sağlayacaktır.

Bu aşamadan sonra, bileşen seviyesindeki modellemeye geçilebilir. Soyut seviyedeki yapıya bağlı kalınarak kullanılacak bileşenlerin modele eklenmesi gerekmektedir. Yaptığımız çalışma, modelden çalışır kod üretmeyi de planlamaktadır, bu nedenle gerçek ortamdaki bileşenlerin modele aktarılabilmesi önemlidir.

İç gözlem mekanizmaları kullanılarak, bileşenlerin dış dünyaya sunduğu arayüz bilgilerine ulaşılabilmektedir. Elde edilen bilgiler, modelleme aracında mevcut bileşenlerin grafiksel olarak gösterimine olanak sağlamaktadır. Aktarımın tamamlanmasından sonra mevcut bileşenler arasındaki ilişkilerin modellenmesine geçilir. Arayüzler arasındaki ilişkilerin, mesaj alışverişlerinin gösterildiği bu aşamada, sistemin bileşen seviyesindeki tüm işlevselliği çözümlenmeye çalışılmalıdır. Model üzerinden yazılım yapmayı amaçladığımızdan, yapılan modellemenin kalitesi, geliştirilecek sistemin kalitesini belirleyecektir.

Bileşen seviyesindeki bağların ve mesaj alışverişlerinin tamamlanması ile COSEML'in önerdiği modele ulaşılmış demektir. Yaptığımız çalışma bu noktadan itibaren bileşenler arasındaki iş mantıklarının ve etkileşimin işbirliği diyagramları ile modellenmesini öngörmektedir.

4. İşbirliği Diyagramlarının Kullanımı

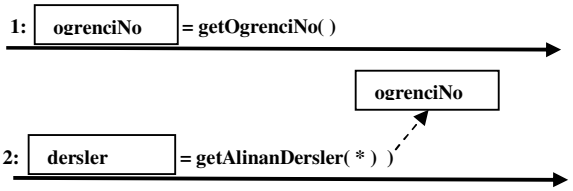
UML deki sınıf diyagramları üzerinden kod üretimi yapılabilmektedir fakat üretilen bu kodlar işlevsellikten uzaktır. Modelde belirtilen özellik ve yöntemleri tanımlama dışında herhangi bir iş mantığı içerememektedir. Ancak işbirliği diyagramlarının durum tabanlı yapısı ve sırasal özelliği fonksiyonel kod üretimine olanak sağlamaktadır [6]. Sınıf diyagramlarının kod üretiminde yetersiz olduğu görülmüştür. Modelde işbirliği diyagramlarının kullanılması kod üretimine önemli katkılar sağlayacaktır. Tasarladığımız işbirliği diyagramları, bileşenler arasındaki etkileşimleri, yöntem çağrımı ve olay dinleme seviyesinde detaylı olarak modellemektedir. Bu diyagramların kullanımında öncelikli amacımız, oluşturulacak modelden bire bir kod üretilebilmesini sağlamaktır. Bunu sağlamak amacıyla, modelleme ortamına gerçek bileşenlerin aktarılması gerekmektedir. Bileşen aktarımı özelliği, mevcut iç gözlem mekanizmaları (COM+ da IDispatch Interface, CORBA da Dynamic Invocation Interface (DII), EJB de Java Reflection, JavaBeans de Introspection vb.) kullanılarak modelleme aracına kazandırılabilir.
Aktarımdan sonra, modelleme işlemi bileşenlerin arayüzleri kullanılarak yapılacaktır. Modele eklenecek olan bileşen arayüzlerdir. Bu kullanım, bileşene ait kullanılmayan arayüzlerinin modelde gereksiz karışıklık yaratmasını engelleyecektir.
Arayüzler arasındaki etkileşim, yöntem çağrımı ve olay dinleme ile gerçekleşmektedir. İşbirliği diyagramlarından kod üretebilmek isteniyorsa parametre alan yöntem yapılarının desteklenmesi gerekmektedir. Bununla birlikte aynı verinin farklı yöntemlere parametre olarak gitmesi durumu da göz önüne alınmalıdır. Bu amaçla, çağrılan yöntemlerden döndürülen nesnelerin grafiksel

gösterimli değişkenlere atanmasını ve bu değişkenlerin model üzerindeki diğer yöntemlerde kesik çizgili oklar vasıtasıyla parametre olarak kullanılabilmesini planlamaktayız.

Diyagramdan gerçek manada kod üretebilmek için mevcut yapısına ek kurallar tanımlanması gerekmedir. Fakat tanımlanan kurallar diyagramın basitliğine ve anlaşılırlığına engel olmamalıdır. İşbirliği diyagramlarından kod üretilmesi konusunda yapılan bazı çalışmalarda [7,8] detaylı kurallar tanımlanmış, bu da modeli çok karmaşık hale getirmiştir. Yaptığımız çalışmada tanımladığımız yöntemlerin, modelin anlaşılırlığını engellememesine dikkat edilmiştir.

5. İşbirliği Diyagramında Değişken Kullanımı

İş birliği diyagramlarında model üzerinde kullanılmak üzere değişken tanımlanabilmektedir. Bu değişkenlere çağrılan yöntemlerin döndürdüğü değerler Şekil 2'de gösterildiği gibi atanabilir. Daha sonra bu değişken model içinde başka yöntemlere parametre olarak kullanılabilir.

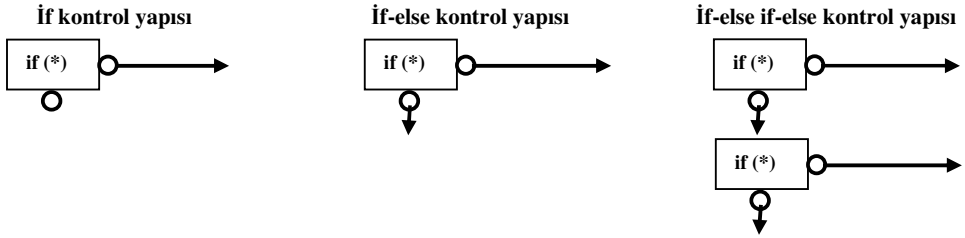


Şekil 2. Değişken atanması ve parametre olarak kullanımı

6. İşbirliği Diyagramında Kontrol Yapılarının Kullanımı

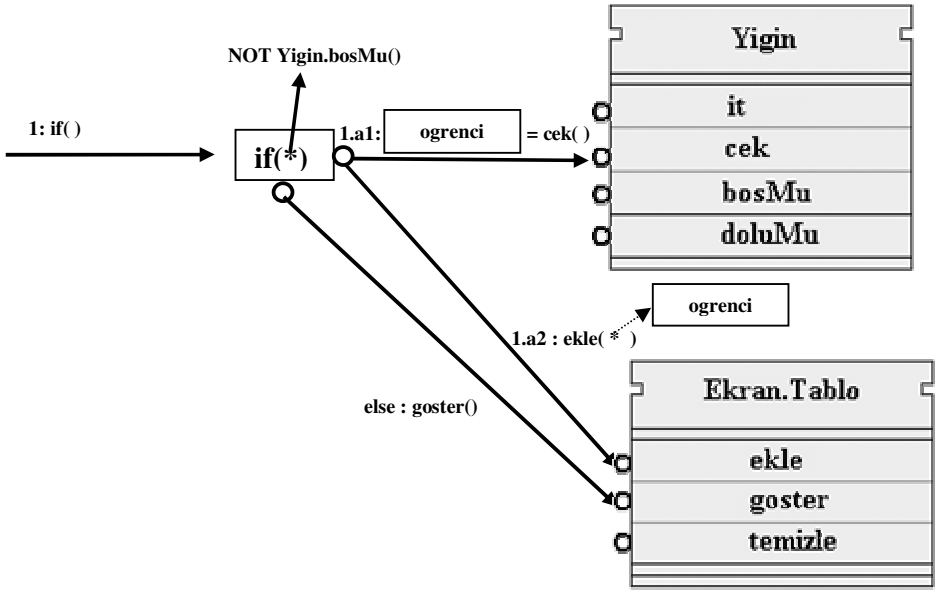
Karmaşık iş mantıklarının modellenebilmesi amacıyla, işbirliği diyagramlarına if/else kontrol mekanizmalarının eklenmesi gerekmektedir, böylece arayüzler arasında koşullu yöntem çağrımı desteklenmiş olacaktır.

Şekil 3'de if kontrol yapısı kullanılarak if, if-else ve if-else if-else yapılarının işbirliği diyagramlarında nasıl modellenebileceği gösterilmiştir.



Şekil 3. Kontrol yapılarının kullanımı

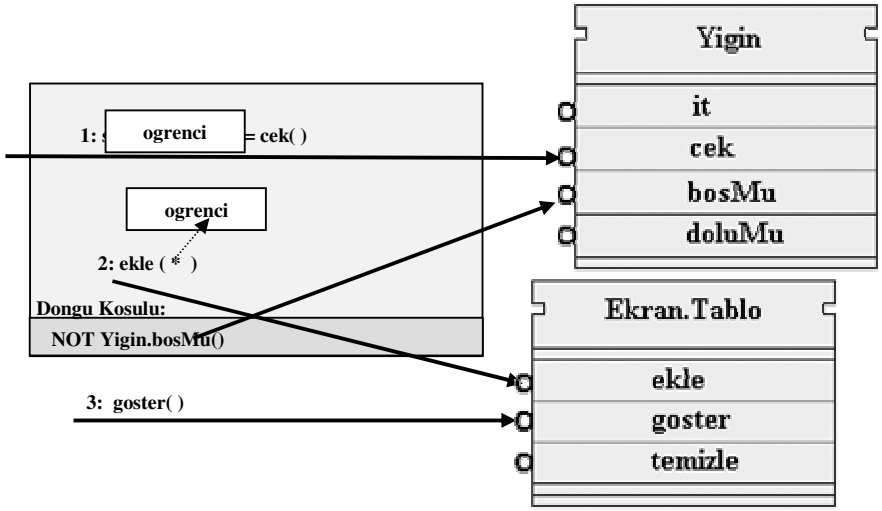
Şekil 4'teki model örneğinde, yığın boş ise ekrandaki tablonun görüntülenmesi, aksi taktirde yığında sıradaki öğrenci nesnesinin tabloya eklenmesi mantığı modellenmiştir.



Şekil 4. if / else kontrol yapısına örnek bir model

7. İşbirliği Diyagramında Döngü Yapılarının Kullanımı

İşbirliği diyagramında döngü yapısını sağlamak amacıyla dikdörtgen blok kullanıldı. Bu bloğun içine alınan işbirliği diyagram adımları, blok altındaki döngü koşulu sağlandığı sürece sırasıyla çağrılmaya devam edecektir. Şekil 5'te döngü yapısına örnek bir kullanım görüntülenmiştir. Şekildeki modelde, yığında *öğrenci* nesneleri varolduğu sürece bu nesneler yığından çekilerek *Ekran* bileşenindeki *Tablo* arayüzüne eklenmektedir. Yığında nesne kalmadığı zaman *bosMu()* yöntemi "doğru" döndürecek ve döngü sona erecektir. Sonraki adımın çalışmasıyla, *Tablo* arayüzüne eklenen *öğrenci* nesneleri görüntülenecektir



Şekil 5. Döngü yapısının kullanımı

8. Örnek İşbirliği Diyagramı

Bu örnekte, İngilizce bir yazının, cep telefonuyla fotoğrafı çekilip gönderildiğinde Türkçe karşılığını mesaj atan bir servisi BYYM ile yapacağımızı varsayalım. Bu servisi COSEML'deki işbirliği diyagramıyla modelleyelim. Yapılacak bu servisle ilgili senaryo alttaki maddelerden oluşmaktadır.

1. Kullanıcı, İngilizce metnin fotoğrafını çektikten sonra servise mesaj atar.
2. Servis mesajı kaydeder, mesajdaki fotoğrafı çevrim yapmak üzere işleme alır.
3. Optik karakter tanıma yöntemi kullanarak fotoğrafı, yazıya dönüştürür.
4. Bu metin İngilizce'den Türkçe'ye çevrilir.
5. Çevrimi yapılmış yazı, bir mesaj oluşturularak kullanıcıya geri gönderilir.

Senaryo incelendikten sonra yapılan araştırmada, piyasadaki üç bileşenin gereksinimleri karşıladığını varsayalım. Bunlardan ilki telefonla gönderilen mesajları alan ve mesaj gönderebilen MobilMesaj bileşeni, ikincisi bir fotoğraftaki karakterleri tanıyarak yazılı metne çevirebilen OCR bileşeni ve sonuncusu bir metni farklı dillere çevirebilen Çevirmen bileşeni olsun.

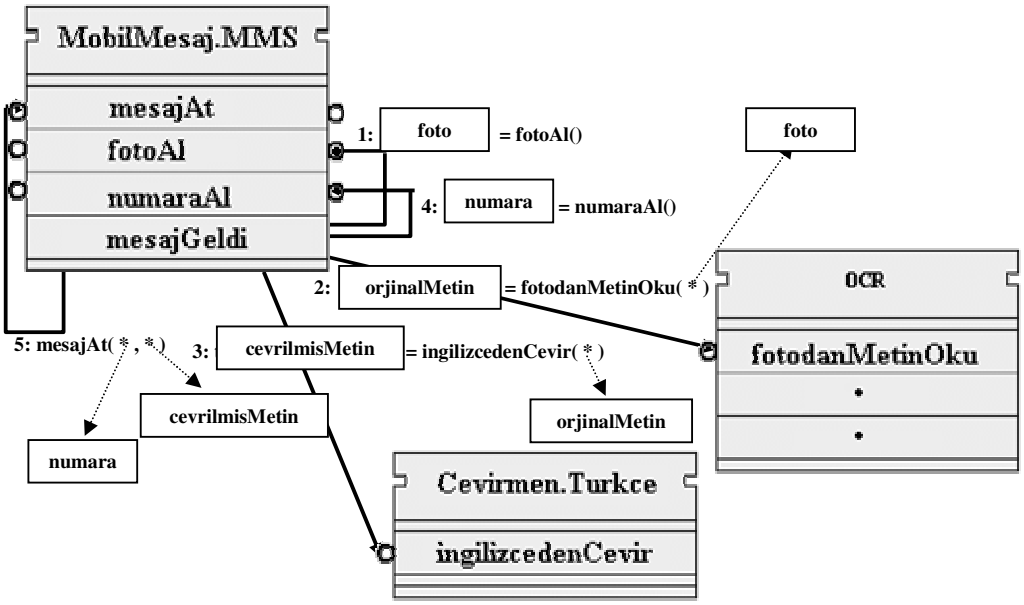
MobilMesaj bileşeninin sadece MMS arayüzünü, Çevirmen bileşenin Turkce arayüzünü ve tek arayüzden oluşan OCR bileşenini kullanarak senaryoyu uygulayabiliriz.

İşbirliği diyagramlarının sıralı çağırmaya dayalı yapısı, bu tür senaryoların modellenmesi ve çalışır kod üretilmesi için çok uygun bir zemin oluşturmaktadır. Bu özellik, senaryoyu kolay ve etkin şekilde modellememize olanak sağlar.

Senaryonun ilk adımında kullanıcı çevrimini yaptırmak istediği fotoğrafı çekerek servise mesaj atmış ve ikinci aşamada mesaj servise ulaşmıştır. Bu durumda MobilMesaj.MMS arayüzündeki mesajGeldi olayı aktif olur. Senaryo bu olayla başlamaktadır bu yüzden işbirliği diyagramımız da

bu olayla başlamalıdır. Üçüncü adımda, gelen mesajdaki fotoğrafın OCR ile metne dönüştürülme işlemi vardır. Bunun için OCR bileşeninin fotodanMetinOku(foto) yöntemi kullanılacaktır. Bu yöntemin gerek duyduğu foto parametresi MobileMessage.MMS arayüzündeki fotoAl() yöntemiyle elde edilebilmektedir. Bu yüzden başlama noktası mesajGeldi() olayı olan modelin 1. adımında fotoAl() yöntemini çağırıp dönen sonucu da foto adında bir değişkene atayabiliriz. 2. adımda OCR bileşeninin fotodanMetinOku(foto) yöntemini 1. adımda atadığımız foto parametresi ile çağırabiliriz. Bu yöntemin döndürdüğü değer sonraki adımlarda kullanılacağından bu yöntemin sonucunu da orjinalMetin adında bir değişkene atayabiliriz. 3. adımda Çevirmen.Turkce arayüzündeki ingilizcedenCevir(metin) yöntemi orjinalMetin parametresi ile çağırıp dönen sonuç için de cevrilmisMetin adında yeni bir değişken kullanabiliriz. Bu adımların ardından geri göndereceğimiz çevrilmiş metin artık elimizde olduğundan MobileMessage.MMS arayüzündeki mesajAt() yöntemini kullanarak çeviriyi müşteriye gönderebiliriz. Bu yöntem telefon numarası ve metin olmak üzere iki parametre almaktadır. 4. adımda yine aynı arayüzün mesajAt () yöntemini çağırıp sonucu numara adında bir değişkene atadıktan sonra 5. adımda mesajAt() yöntemini numara ve cevrilmisMetin parametreleriyle çağırdığımızda servisin modellemesini tamamlamış oluruz.

Şekil 6'te bu adımların işbirliği diyagramı ile modellenmiş hali görülmektedir.



Şekil 6. Mesajla çevrim yapan GSM servisinin işbirliği diyagramı modeli

Bu model tarafından üretilecek muhtemel Java kodu da alttaki gibi olacaktır.

```
mesajGeldi (mms)
{       OCR ocr = new OCR();
        Cevirmen.Turkce cevirmen = new Cevirmen.Turkce();
        ByteArray foto = mms.fotoAl();
        String orjinalMetin = ocr.fotodanMetinOku(foto);
        String cevrilmisMetin = cevirmen.ingilizcedenCevir(orjinalMetin);
        String numara = mms.numaraAl();
        mms.mesajAt(numara, cevrilmisMetin ()
}
```

9. Sonuç

Standart modelleme dili olarak kabul edilen UML, nesneye yönelik yazılım modellemelerinde çok başarılı olmasına ve yaygın olarak kullanılmasına rağmen BYYM paradigmasının modelleme ihtiyacına cevap verememektedir. Bu durum modelleme konusunda bir boşluk oluşturmuş ve bu boşluğu doldurmak amacıyla BYYM perspektifiyle COSEML tasarlanmıştır. Bu modelleme dili gelişim aşamasındadır. Yaptığımız çalışmayla işbirliği diyagramlarının COSEML'e dahil edilmesi önerilmiş böylece modelleme dilinin akademik ve profesyonel kapsamını ve kullanılabilirliğini arttırmak amaçlanmıştır. Gerek modelin sistemi daha iyi anlatmasına olanak sağlaması, gerekse modelden çalışır kod üretilebilmesine çok uygun bir ortam sağlaması bakımından sunduğumuz çalışmanın COSEML'e önemli katkılar sağlayacağı inancındayız.

Kaynakça

[1]. UML Group: Unified Modeling Language 1.1. Rational, 1997

[2]. M. M. Tanik ve A. Ertas, "Interdisciplinary Design and Process Science: A Discourse on Scientific Method for the Integration Age", Journal of integrated Design and Process Science, September 1997, s. 76-94.

[3]. K. Bergner, A. Rausch, M. Sihling, "Componentware – The Big Picture", Technische University, München, 1998.

[4]. Ali H. Dogru, "Component-Oriented Software Engineering Modeling Language: COSEML", Aralık 1999, Orta Doğu Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, Türkiye.

[5]. Aydın Kara, A Graphical Editor For Component Oriented Modeling. Yüksek Lisans tezi, Nisan 2001, Orta Doğu Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, Türkiye.

[6]. Engels G., Hucking R., Sauer S., Wagner A., "UML Collaboration Diagrams and their Transformation into Java.", In France R. and Rumpe B. (Ed.), Proceedings of UML'99, Lecture Notes in Computer Science 1723 (s. 473–488), Colorado, USA, 1999.

[7]. McLeish K., Code Generation from UML Models using XMI. Unpublished manuscript, Colorado State University, USA, 2001.

[8]. Dinh T. T. T., Rules For Generating Code From Uml Collaboration Diagrams And Activity Diagrams, Yüksek Lisans Tezi, Colorado State University, Computer Science Department, USA, 2003.