BİR BİLEŞEN KÜTÜPHANESİNDEKİ YENİDEN KULLANILABİLİR JAVA BİLEŞENLERİNİN SINIFLANDIRILMASI ve BİLEŞEN ÜST-BİLGİLERİNİN TANIMLANMASI

Mustafa TÜRKSEVER*
R.Cenk ERDUR *

ÖZET

Günümüzde Java ile platform bağımsız bileşenler yazmak, bu bileşenleri yeniden kullanılabilir bileşen kütüphanelerinde saklamak ve İnternet üzerinden kullanıma sunmak olasıdır. Java'nın "Birkere yaz heryerde çalıştır" felsefesi ile yeniden kullanım için gerekli olan altyapı sağlanmış durumdadır. Bu altyapının sağlanması ile yeniden kullanıma dayalı yazılım geliştiren kişilerin çeşitli uygulama alanları için yazılmış Java bileşen kütüphanelerine gereksinimi giderek artacaktır. Bu çalışmada, üç farklı uygulama alanında Java bileşenleri içeren bir kütüphane geliştirilmiştir. Bileşen kütüphaneleri ile ilgili diğer önemli bir konu da sınıflandırma ve üst-bilgi tanımlamadır. Eğer bileşenlerin kolay ve etkin bir şekilde aranabilmesi, bulunabilmesi ve anlaşılabilmesi isteniyorsa, kütüphanedeki bileşenlerin sınıflandırılması ve bileşenler için üst-bilginin tanımlanması gerekmektedir. Bu çalışma kapsamında geliştirilen kütüphanedeki bileşenler boyutlara ayırmaya dayanan yaklaşım ile sınıflandırılmıştır. Bileşen üst-bilgileri ise RIG-BIDM standardının bir alt kümesi kullanılarak tanımlanmıştır.

SUMMARY

Classification and meta-knowledge definition for reusable Java components in a component library.

Today it is possible to write platform independent Java components, to store the produced components in libraries, and to market them over Internet. Java with the slogan "write once, run everywhere" has provided the Java infrastructure for Internet based software reuse. In a near future, there will be thousands of component libraries providing components in different domains, and

Ege Üniversitesi Bilgisayar Mühendisliği Bölümü - Bornova-İZMİR

reuse based software developers will develop software by integrating the components that they have downloaded from different libraries. In this study, a Java component library providing components in three different domains have been developed. Another important issue about component libraries is classification of components and definition of component meta-knowledge. In this study faceted classification scheme has been used for classification of components, and the RIG-BIDM model has been used for meta-knowledge definition.

1. GİRİŞ

Yeniden kullanım, gerek günlük hayatta gerekse bilimsel alanlarda karşımıza çıkan genel bir kavramdır. Aynı tür tuğlaların değişik binaların yapılışında kullanımı, mimarların aynı tasarım yöntemlerini değişik projelerde uygulaması, fizikçilerin aynı fizik kurallarını farklı problemlerin çözümünde kullanması, bilgisayar veya elektronik mühendislerinin tümleşik devrelerle mikro-işleyicili sistemler tasarlaması yeniden kullanım örnekleridir (Erdur, 1995).

Yazılım mühendisliği alanında yeniden kullanım, önceki yazılım geliştirmelerde üretilen bileşenlerin yeni yazılımların geliştirilmesinde kullanılmasıdır. Yazılım geliştirmenin maliyeti ve süresi üzerinde yapılan istatistikler, yeni baştan yapılan tasarımları ve kodlamaları maliyet ve süreyi artıran en önemli unsurlar olarak göstermektedir (Griss, 1993). Bu durumda getirilebilecek en iyi çözümlerden birisi yeniden kullanıma dayalı yazılım geliştirme olmaktadır. Yeniden kullanım, yazılan kod miktarını azalttığı için maliyeti düşürmekte ve süreci hızlandırmaktadır. Ayrıca,yeniden kullanılabilir bileşenler daha önceden sınanmış ve belli bir standarda uyumlu olduğu için geliştirilen yazılımların niteliği de artmaktadır.

Günümüzde Java ile platform bağımsız bileşenler yazmak ve bu bileşenleri İnternet üzerinden ücretsiz kullanıma sunmak veya pazarlamak olasıdır. Java'nın "Birkere yaz heryerde çalıştır" felsefesi ve JavaBeans bileşen modeli ile yeniden kullanım için gerekli olan altyapı sağlanmış durumdadır. Yeniden kullanıma dayalı yazılım geliştirenler, İnternet üzerinden elde ettikleri Java uyumlu bileşenleri görsel uygulama geliştirme araçları yardımı ile geliştirmekte oldukları yazılımlara katabileceklerdir. Bu yöntem oldukça hızlı ve güvenilir bir uygulama geliştirme yöntemi olarak kabul görmektedir. Bu nedenlerle, yeniden kullanıma dayalı yazılım geliştiren kişilerin kullanımına sunulmak üzere çeşitli uygulama alanları için yazılmış Java bileşenleri içeren kütüphaneler tasarlanmalıdır.

Bu çalışma kapsamında, Java Beans bileşen modeline uyumlu bileşenler içeren bir kütüphanenin gerçekleştirimi anlatılmaktadır. Oluşturulan kütüphane temel veri yapıları ve algoritmalar ile matematiksel hesaplamalar alanlarında bazı bileşenler içermektedir.

Hangi teknoloji ile geliştirilmiş olursa olsun, bir yeniden kullanılabilir bileşenler kütüphanesinin etkin olarak aranabilmesi ve elde edilen bileşenlerin kolaylıkla anlaşılabilmesi için kütüphanedeki bileşenlerin sınıflandırılması ve bileşenler için üst-bilginin (meta-knowledge) tanımlanması gerekmektedir. Bu çalışma kapsamında geliştirilen kütüphanedeki bileşenler boyutlara ayırmaya dayanan (faceted) yaklaşım ile sınıflandırılmıştır. Bileşenlere ilişkin üst-biligler RIG-BIDM (Reuse Library Interoperability Group - Basic Interoperability Data Model) standardının bir alt kümesi kullanılarak tanımlanmıştır. Sınıflandırma bilgileri ve bileşenlere ilişkin üst-bilgiler ilişkisel veri tabanında saklanmıştır. Bu bilgiler ileride bileşenlere erişimde kullanıcılara kolaylık sağlayacaktır.

Bundan sonraki bölümde, yazılım mühendisliğinde yeniden kullanım alanında yapılan çalışmalar ile ilgili genel bir literatür taraması verilmekte, boyutlara ayırmaya dayanan sınıflandırma yaklaşımı ile bileşen üst-bilgileri tanımlamda kullanılan RIG-BIDM standardı tanıtılmaktadır.

Üçüncü bölümde oluşturulan Java bileşen kütüphanesi tanıtılmakta ve bileşenlerin sınıflandırılması ve üst-bilgilerin hazırlanması örnekler üzerinde tartışılmaktadır. Dördüncü bölümde çalışmadan elde edilen sonuçlar verilmektedir. Yararlanılan kaynakların listesi beşinci bölümde yer almaktadır.

2. YAZILIM MÜHENDİSLİĞİNDE YENİDEN KULLANIM

Yeniden kullanımın yazılım mühendisliği alanında uygulanması ile ilgili ilk düşünceler, 1968 yılında McIllroy tarafından ortaya atılmıştır (Prieto-Diaz and Freeman, 1987; Krueger, 1992). McIllroy, nümerik hesaplamalar, giriş/çıkış işlemleri, metinsel işlemler ve dinamik bellek yönetimi konularında yeniden kullanılabilir fonksiyonlar hazırlanmasını, bu fonksiyonların kaynak kodlarının bir kütüphanede, tanımlarının ise kataloglarda saklanmasını önermiştir. Böylece, kataloglardan seçilen fonksiyonların kütüphanedeki kaynak kodlarının yeni bir yazılım geliştirme sürecinde kullanılabilmesi olanağı doğmuştur.

McIllroy'dan sonra, 1974 yılında, 'yazılım fabrikası' fikri ortaya atılmıştır. Sadece yeniden kullanılabilir fonksiyonlar (kaynak kod) içeren bir kütüphanedeki fonksiyonların yeni yazılım geliştirme süreçlerinde kullanımı için kullanıcının analiz ve tasarım çalışmalarını önceden tamamlaması gerekmektedir çünkü tasarımın ihtiyaç duyduğu fonksiyonlar - eğer kütüphanede mevcutsa - seçilecektir. Analiz ve tasarımların , mevcut yeniden kullanılabilir yapılar gözönüne alınarak ve yeniden kullanılabilir soyutlamalardan (analiz, tasarım, dokümanlar) da yararlanarak gerçekleştirilmesinin daha iyi bir yöntem olduğu görünmektedir. Yazılım fabrikası fikri, bu yöntemi desteklemek üzere ortaya atılmıştır. Bu yöntem ile, yazılım geliştirme süreci ve stili değişmektedir. Bu tip bir stili destekleyici organizasyonlara yazılım fabrikası (Griss, 1993) denmektedir. İlk olarak, 'Systems Development Corporation' isimli şirket tarafından kullanılan bu fikir daha sonraları özellikle Japonya'da birçok kuruluş tarafından temel alınarak belli bir sistematik çerçevesinde geliştirilmiştir.

Yazılım fabrikası fikri, yazılım geliştiricilerin sadece kaynak kod değil, aynı zamanda her türlü eski ürünü (analiz, tasarım, dokümantasyon vb.) belli kurallar çerçevesinde kullanılmasını desteklediği için, McIllroy'un kaynak kod kütüphanesi fikrinin geliştirilmiş bir şekli olarak nitelendirilmektedir.

Yazılım mühendisliğinde yeniden kullanım ile ilgili akademik çalışmalar 1970'li yılların sonlarında başlamıştır. Araştırma projeleri ile ilgili sonuç raporları 1983 yılında toplanan 'Programlamada Yeniden Kullanım' isimli ilk uluslarası yeniden kullanım kongresinde tartışılmıştır (Biggerstaff and Perlis, 1984; Prieto-Diaz, 1993).

Yeniden kullanılabilir bileşenler ile yazılım geliştirme son yıllarda yazılım mühendisliği alanının en yoğun çalışılan konularından birisi durumuna gelmiştir. Yeniden kullanım alanındaki araştırma konuları arasında birçok konunun yanısıra aşağıdaki başlıklar da yer almaktadır.

i. Yeniden kullanılabilir bileşen kütüphaneleri tasarımı.
ii. Kütüphanelerdeki bileşenler ile ilgili üst bilgilerin (meta-knowledge) belirlenmesi.
iii. Bileşen kütüphanelerine gerek yerel gerekse Internet üzerinden etkin erişimi sağlayan araçların tasarımı, bulunan kaynak kod içeren bileşenlerin birleştirilmesini sağlayacak araçların tasarımı.
iv. Bilinen yazılım geliştirme süreçlerinin yeniden kullanıma dayalı bir biçime dönüştürülmesi.

REBOOT (Morel and Faget, 1993), SODALIA (Mambella et al., 1995), STARS (Davis, 1992) bu konuda yürütülen önemli çalışmalara örneklerdir. Bunun dışında RIG (Reuse Library Interoperability Group-Yeniden Kullanılabilir Kütüphane İçişletilebilirlik Grubu) grubu bileşenlerin üst-bilgilerinin modellenmesinde kullanılabilecek çalışmalar yapmaktadır (Hobbs, 1993).

Bu çalışma kapsamında yukarıda i ve ii numaralı maddelerde belirtilen araştırma konuları üzerinde durulmaktadır.

2.1. Boyutlara Ayırmaya Dayanan Sınıflandırma Yaklaşımı

Yeniden kullanıma dayalı yazılım geliştirmede en önemli problemlerden birisi, binlerce bileşenden oluşan bir kütüphaneden istenilen özellikte bileşenlerin seçilmesidir. Bu soruna bir çözüm getirebilmek için, kütüphanedeki bileşenlerin belli bir sınıflandırma tasarısı kullanılarak sınıflandırılması gerekmektedir. Boyutlara ayırmaya dayanan (faceted) sınıflandırma tasarısı (Prieto-Diaz and Freeman, 1987) yeniden kullanılabilir bileşenlerin sınıflandırılması amacıyla en çok kullanılan sınıflandırma yöntemlerinden birisidir ve aşağıda tanıtılmaktadır.

Boyutlara ayırmaya dayanan sınıflandırma yaklaşımı, bilginin sınıflandırma işleminde kullanılabilecek farklı niteliklerini gözönüne almaktadır. Her farklı nitelik bir boyut ile gösterilmektedir. Her boyutta, temsil ettiği nitelik ile ilgili birtakım terimler (anahtar kelimeler) bulunmaktadır. Boyutların içindeki terimler, o boyuta ilişkin terimler uzayını oluşturmaktadır. Bu yaklaşımda, bir bileşen farklı boyutlardan seçilen terimlerin birleştirilmesinden oluşan tanımlayıcı (anahtar) ile sınıflandırılmaktadır.

Boyutlara ayırmaya dayanan yaklaşımın daha iyi anlaşılabilmesi için aşağıda bazı tanımlamalar yapılarak, bu tanımlamalara göre bileşen tanımlayıcılarının nasıl oluşturulduğu gösterilmektedir.

B_i : i numaralı niteliğe ilişkin boyutu göstermektedir. Diğer bir deyişle, i numaralı nitelik üzerinde tanımlı terimlerin tümünün kümesidir.

T_{ij} : i numaralı niteliğe ilişkin boyuttaki j numaralı terime karşılık gelmektedir.

Buna göre $B_i=\{T_{ij} \mid k \geq j \geq 1\}$ (k : i numaralı niteliğe ilişkin boyuttaki terim sayısı) biçiminde tanımlanmaktadır.

Örnek olarak, yazılım bileşenlerinin 3 nitelik ile tanımlandığı bir tasarıda, bir yazılım bileşenine ilişkin tanımlayıcı her niteliğe ilişkin boyuttan alınacak birer terimden oluşacak ve $<T_{1x}, T_{2y}, T_{3z}>$ biçiminde olacaktır. Burada, x, y ve z ilgili boyuttaki terim numaralarını göstermektedir.

Daha somut bir örnek vermek gerekirse, niteliklerin ve bu niteliklere ilşkin boyutların şekil-1'deki gibi olduğu bir durumda, Unix ortamında bir ağaç oluşturan bileşeni sınıflandırmak için <oluştur, ağaç, Unix> şeklinde bir tanımlayıcı (anahtar) oluşturmak gerekmektedir.

B1={ ekle, oluştur, dolaş, ... } (bileşen işlevi niteliği)

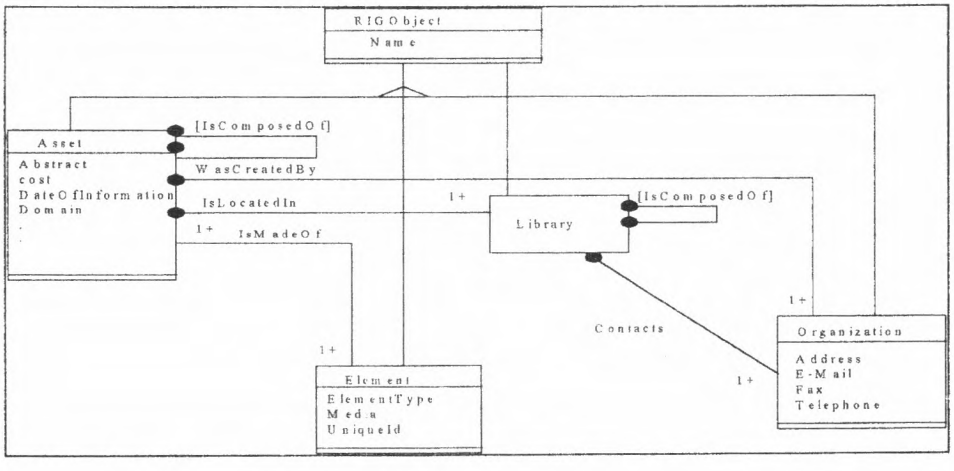
B2={ ağaç, dizin, ... } (üzerinde işlem yapılan nesneler)

B3={ Dos, Unix, ... } (sistem türü)

Şekil-1. Sınıflandırma Boyutlarına Örnek

2.2. RIG-BIDM Standardı ve Bileşen Üst-Bilgisi Tanımlama

Bileşenlere ilişkin ek açıklayıcı bilgiler (bileşenin sürümü, yazılma tarihi, yazarı, yazıldığı kuruluş, kuruluş adres bilgileri vb.) olan bileşen üst-bilgisinin (meta-knowledge) yeniden kullanıma dayalı yazılım geliştirmede önemli bir yeri bulunmaktadır. Bunun nedeni, bileşen üst-bilgisinin yeniden kullanıcılara bileşen seçiminde büyük kolaylık sağlayabilmesidir. Bu çalışmada bileşen üst-bilgisi RIG-BIDM (Reuse Library Interoperability Group-Basic Interoperability Data Model) temel alınarak tanımlanmıştır. RIG, yeniden kullanılabilir bileşen kütüphanelerinin içten-işletilebilirliğini (interoperability) sağlamaya yönelik standartlar geliştirilmesi amacıyla 1991 yılında kurulmuş bir çalışma grubudur (Browne and Moore, 1997). BIDM ise yeniden kullanılabilir bileşen kütüphanelerinin içten-işletilebilirliğinin sağlanması için bu kütüphanelerin birbiri ile değişeceği bilgileri tanımlayan bir veri modelidir. BIDM ile, kullanıcılar tek bir arayüz kullanarak farklı bileşen kütüphanelerindeki bileşenlere erişme olanağına kavuşmaktadır.BIDM veri modeli, belli bir sıradüzende olan sınıflardan oluşmaktadır. Bu sınıflara ilişkin nitelikler de (attributes) belirlenmiştir. BIDM'in, James Rumbaugh'ın nesneye yönelik modelleme ve tasarım metodolojisinin grafiksel notasyonunda gösterimi Şekil-2.'de verilmektedir (Browne and Moore, 1997).



Şekil-2. BIDM Veri Modeli (Browne and Moore, 1997'den)

Bu çalışmada bileşen üst-bilgileri yukarıda sözedilen BIDM veri modelinin bir alt kümesi kullanılarak tanımlanmıştır. Tanımlanan üst-bilgi modelinin hangi sınıflardan ve niteliklerden oluştuğu dördüncü bölümde anlatılmaktadır.

3. OLUŞTURULAN JAVA BİLEŞENLERİ KÜTÜPHANESİNİN TANITIMI

Oluşturulan kütüphane aşağıda listelenen kaynak kod bileşenleri içermektedir.

Veri Yapıları Uygulama Alanı:

Doğrusal veri yapıları alanında aşağıdaki bileşenler bulunmaktadır.

- Yığıt işlemleri içeren sınıf.
- Kuyruk işlemleri içeren sınıf.
- Bağlaçlı liste işlemleri içeren sınıflar.

Doğrusal olmayan veri yapıları alanında aşağıdaki bileşenler bulunmaktadır.

- Ağaç işlemleri içeren sınıflar.

Genel Algoritmalar Uygulama Alanı:

Arama algoritmaları alanında aşağıdaki bileşenler bulunmaktadır.

- İkili arama yapan sınıf.
- Sıradan arama yapan sınıf.

Sıralama algoritmaları alanında aşağıdaki bileşenler bulunmaktadır.

- Kabarcık sıralama yapan sınıf.
- Seçmeli sıralama yapan sınıf.
- Eklemeli sıralama yapan sınıf.

Çokluortam alanında aşağıdaki bileşenler bulunmaktadır.

- Görüntü yükleyen sınıf.
- Ses yükleyen ve çalan sınıf.

Bazı Arayüzler İçeren Bileşenler aşağıdakilerdir.

- Düğme işlemleri ile ilgili arayüz sınıf.
- Fare işlemleri ile ilgili arayüz sınıf.
- Klavye işlemleri ile ilgili arayüz sınıf.

Matematik Uygulama Alanı:

- Matris işlemleri yapan sınıf.
- Kompleks sayılar üzerinde işlem yapan sınıf.
- Kesir işlemleri sınıfı.

Yukarıda görüldüğü gibi bileşenler 3 farklı uygulama alanı için yazılmıştır. Boyutlara ayırmaya dayanan sınıflandırma yaklaşımının bu uygulama alanlarına uygulanması sonucunda aşağıda belirtilen boyutlar belirlenmiştir.

Boyut-1: Soyutlama

Bu boyut ilgili uygulama alanı içindeki konuları ifade etmek için kullanılmaktadır. Örneğin, yığıt veya matris bu boyuta ili°kin terimlere örneklerdir.

Boyut-2: İşlem

Bu boyut ilgili soyutlama üzerinde uygulanabilecek işlem türlerini belirlemektedir. Örneğin, yığıt üzerinde "push" işlemi gibi.

Boyut-3: Tür

Bu boyut yeniden kullanılabilir bileşenin türünü belirlemektedir. Yeniden kullanılabilir bileşenler yazılım mühendisliği yaşam döngüsünün her ürünü (analiz, tasarım, kod, belgeleme, vb.) olabilmektedir.

Boyut-4: Dil

Bu boyut bileşenin gerçekleştiriminin yapıldığı dili belirler.

Boyut-5: Ortam

Bu boyut bileşenin işletileceği platformu belirlemektedir.

Buna göre her boyut içindeki terimler aşağıdaki gibi belirlenmiştir. Yeni bileşenler eklendikçe boyutlara yeni terimler eklenebilir veya çıkartılabilir.

Soyutlama	İşlem	Tür	Dil	Ortam
Yığıt	push	kod	java	win95
Kuyruk	pop	analiz		unix
Bağlaçlı-Liste	ekle_başa	tasarım		
Ağaç	sil_baştan	belgeleme		
Sıralama	preorderdolaş			
Arama	inorderdola°			
Matris	postorderdola°			
Kopleks	kabarcık			
Kesir	eklemeli			
Çokluortam	seçmeli			
Arayüz	sıradan			
	ikili			
	topla			
	çıkar			

Soyutlama	işlem
	çarp
	böl
	ekle_sona
	sil_sondan
	ekle
	ses_yükle
	görüntü_yükle
	fare_i°lem
	klavye_i°lem
	düğme_işlem

Bileşenlere ilişkin üst-bilgiler ise BIDM standardının aşağıda belirtilen alt kümesi temel alınarak hazırlanmıştır.

Üst-bilgi içeriği:

- Sınıflama bilgisi (boyutlara ayırmaya dayanan yaklaşıma göre bileşeni belirleyen terimlerden oluşan anahtar.)
- Bileşen sürümü
- Bileşenin yazıldığı organizasyon
- Bileşenin yazıldığı tarih
- Ücret (Ticari amaçlı ise)
- Bileşeni tanıtıcı kısa bilgiler
- Bileşenin kullanımına ilişkin kısıtlamalar
- Bileşeni daha önceden kullanmış kişilerin deneyimleri
- Yazar
- E-mail adresi
- Fax/Telefon

Buna göre, örneğin, temel veri yapıları uygulama alanındaki kuyruktan silme yapan bir bileşene ilişkin üst-bilgi aşağıdaki gibi olabilir.

Sınıflama bilgisi: Kuyruk+Sil+Kod+Java+Win95

Sürüm: S.1.1
Organizasyon: Ege Üniversitesi
Tarih: 20/10/1999
Ücret: Yok (Genel amaçlı kullanıma açık)
Kısa Bilgiler: Bu bileşen kuyruğun başındaki elemanı siler.
Kısıtlamalar: Kuyruk içinde tamsayı değerler olmasını gerektirir.
Önceki deneyimler (varsa): Bu bileşen 25/10/1998 tarihinde kullanılmış ve maksimum kuyruk uzunluğunun 20 olması sınırlayıcı bir durum olarak görülmüştür. Bunun için sözkonusu silme metodunu içeren sınıfın içindeki max_len adındaki değişkenin değeri artırılmalıdır.
Yazar: M. Türksever
E-mail: turksever@bornova.ege.edu.tr
Fax/Tel: 3399405 / 3887221-226

4. SONUÇLAR

Java dilinin platform bağımsız bir dil olması nedeni ile yeniden kullanılabilir bileşen geliştirmek için uygunluğu görülmüştür. Windows 95 ortamında geliştirilen kütüphanedeki bileşenler, Unix ortamındaki kullanıcılar tarafından da hiçbir değişiklik yapılmadan kullanılabilmektedir. Java dilinin nesneye dayalı bir dil olması da yeniden kullanımı destekleyen diğer bir özelliğidir. Bir bileşenden kalıtım yolu ile yeni bileşenler türetilebilmekte veya bileşene yeni metodlar eklenerek bileşen kullanıcının isteklerine uyarlanabilmektedir.

Bu çalışmada elde edilen diğer bir sonuç da boyutlara ayırmaya dayanan sınıflandırma yaklaşımının kütüphanedeki dinamik değişimleri karşılayabilecek nitelikte olduğunun görülmesidir. Kütüphaneye yeni bir bileşen eklendiğinde veya bir bileşen çıkarıldığında ilgili boyutlara terim eklemek veya ilgili boyutlardan terim çıkarmak yeterlidir. Bu gibi durumlarda sınıflandırma işleminin baştan yapılmasına gerek olmamaktadır. Örneğin, Algoritmalar uygulama alanında animasyon yapan bir bileşenin kütüphaneye eklendiği düşünülürse, bu bileşen (çokluortam, animasyon, kod, java, win95/unix) tanımlayıcısı ile sınıflandırılacaktır. Buna göre eldeki boyutlar ve terimler gözönüne alındığında (bakınız, bölüm-4) sadece ikinci boyuta animasyon teriminin eklenmesi yeterli olacaktır.

Bileşen üst-bilgilerinin tanımlanması ile bileşenlerin anlaşılmasının kolaylaştığı görülmüştür. Böylece, kullanıcılar ilk önce bileşen üst-bilgisini inceleyerek gereksiz bileşenlerin kendi ortamlarına transferini engelleme şansına sahip olmaktadır. Bu da bilgisayar ağı trafiğine olumlu katkıda bulunacak bir unsurdur.

Bu çalışma çerçevesinde oluşturulan bileşen kütüphanesi gelişmeye açıktır. Yeni Java bileşenleri üretildikçe kütüphaneye eklenecektir. Bileşen kütüphanesi proje bilgisayarı üzerinde paylaşımlı bir dizine konarak Bilgisayar Mühendisliği bölümünde kullanıma sunulacak böylece yeniden kullanıma dayalı yazılım geliştirme kültürü oluşturulmaya çalışılacaktır.

5. KAYNAKLAR

Alden DeSoto, (1997), "Using the Beans Development Kit 1.0", February, Javasoft, USA.

Biggerstaff, T.J. and Perlis, A.J., **(1984), "Foreword", IEEE Transactions on Software Engineering, Se-10(5):474-477.**

Browne, S. and Moore, J., **(1997), "Reuse Library Interoperability and the World Wide Web, in symposium on Software Reusability", ACM Press, USA.**

Davis M., **(1992), "STARS Reuse Maturity Model: Guidelines for Reuse Strategy Formulation", in the proceedings of the fifth Annual Workshop on Software Reuse.**

Deitel, H.M. and Deitel, P.J., (1997), "Java How To Program", Prentice-Hall, USA.

Erdur, R.C., **(1995), "Geniş Ölçekli Yeniden Kullanım Ortamlarında Bileşen Yönelikli Yazılım Geliştirme", Yüksek Lisans Tezi, E.Ü. Fen Bilimleri Enstitüsü, İzmir.**

Griss, M. L., (1993), "Software Reuse from Library to Factory", IBM Systems Journal, 32(4):548-566.

Hobbs E., **(1993), "A Uniform Data model for Reuse Library Interoperability", Columbia, USA.**

Krueger, C.W., **(1992), "Software Reuse", ACM Computing Surveys, 24(2):131-183.**

Mambella E., Ferrari R., De Carli F., Lo Surdo A., **(1995), "An Integrated Approach to Software Reuse Practice", in the proceedings of the Symposium on Software Reusability (SSR'95), ACM Press, Special Issue, USA.**

Morel, J. and Faget, J., (1993), "The Reboot Environment", in Advances in Software Reuse, IEEE Computer Society Press, Los Alamitos, California, 204p.

Prieto-Diaz, R. and Freeman, P., **(1987), "Classifying Software for Reusability", IEEE Software, January:6-16.**

Prieto-Diaz, R., **(1993), "Status Report: Software Reusability", IEEE Software, May:61-66.**