JAWIRO : Java Için Bir Rol Modeli

Yunus Emre Selçuk, Nadia Erdogan

Elektrik-Elektronik Fakültesi

Bilgisayar Mühendisligi Bölümü

Istanbul Teknik Üniversitesi

34469, Ayazaga, Istanbul

e-posta: selcukyu@itu.edu.tr , erdogan@cs.itu.edu.tr

Özet. *Bu çalisma Java dilini rol destegi ile genisletmek üzere gerçekledigimiz JAWIRO adli rol modelini tanitmaktadir. JAWIRO rollerden beklenen temel özelliklerin yanisira literatürde yer alan rol modellerinde tümüyle gerçeklenmemis olan ek özellikleri de herhangi bir kisitlama getirmeden desteklerken, sinif düzeyinde kalitima denk basarim sunar. JAWIRO'yu diger rol modellerinden ayiran ana özelligi ise nesne düzeyinde çoklu kalitimi desteklemesidir.*

1 GIRIS

Gerçek dünya sürekli degisen ve evrimlesen nesnelerden olusmasina ragmen, sinif tabanli olan nesneye yönelik programlamada (NYP) bir nesne ve ait oldugu sinif arasindaki iliski kalici, duragan ve dislamali bir yapidadir. Bu yapi sadece modellenecek gerçek dünya varliklarinin ayrik siniflara bölünebilecegi ve varliklarin siniflarini hiç degistirmeyecegi durumlar için tam anlamiyla uygundur. Bu durum küçük bir elverissizlikten öte, ciddi bir sorun olusturur çünkü dinamik olarak degisen varliklarin statik nesnelerle modellenmesi dikte edilmektedir. Dinamik olarak evrimlesen nesnelerin daha iyi modellenmesi için daha iyi yollar bulma gereksinimi, arastirmacilarin prototip tabanli diller (Ungar, 1987), dinamik siniflandirma (Drossopoulou, 2002) ve özneye yönelik programlama (Wong, 1997) gibi degisik öneriler sunmalarina neden olmustur.

NYP, modellenen sistemdeki her ayri rolün davranisini belirleyen siniflarin tanimlanmasini gerektirir. Nesnelerin birden fazla rol kazanmak yoluyla evrimlestigi sistemlerde, ilgili rollerin her olasi birlikteligi için ilgili siniflardan kalitim yolu ile ek siniflar olusturulmasi gereklidir. "Arakesit siniflari" olarak adlandirilan bu siniflar genellikle çoklu kalitim ile elde edilir. Böylece üstel olarak büyüyen bir sinif hiyerarsisi agaci olusur. Üstel olarak artan sinif sayisina karsilik çogunlukla hiyerarsinin göreli olarak az sayida üyesinden örnekler belli bir anda etkindir. Üstelik bazi nesneye yönelik dillerde çoklu kalitim desteklenmemektedir. Böyle dillerde gerekli islevselligi çok sayida arakesit sinifina kazandirmak gayet zor olacaktir.

Bu çalismada, dinamik olarak evrimlesen gerçek dünya sistemlerini daha iyi modellemek üzere gerçekledigimiz, Java dilini rol destegi ile genisleten bir rol modeli olan JAWIRO incelenmektedir. JAWIRO rollerden beklenen temel gereksinimleri karsiladigi gibi, simdiye dek önerilen rol modellerinin hiçbirinde tümüyle gerçeklenmemis ek islevleri bir arada sunar. Gerçeklenen islevler herhangi bir kisitlama olmaksizin ve basarimdan ödün vermeksizin kullanilabilir.

2 ROL TABANLI PROGRAMLAMA VE ROL MODELLERI

Rol kavrami tiyatro literatüründeki "bir aktörün sahnede oynadigi bir kisim" anlamindaki tanimdan gelmektedir. Roller, degisik türden varliklarin yapabilecegi degisik davranis biçimleridir. Bir varlik bu rollerin bazilarini es zamanli olarak oynayabilir, bazilarini geçici olarak birakabilir, zaman içinde yeni roller kazanabilir veya terk edebilir. Dolayisiyla bir varligi tam anlami ile karakterize eden, bu varligin ilgilendigi rollerin tümüdür.

Roller, Kristensen tarafindan su sekilde tanimlanir: "Bir nesnenin bir rolü, bu nesnenin diger nesnelerin bekledigi gibi davranabilmesi için gereken bir özellikler kümesidir" (Kristensen, 1996). Bir "rol modeli" ise rollerin tasarimi için bir stil ve kullanimi için çesitli islevler sunar. "Rol tabanli

programlama" (RTP), NYP'nin dinamik sistemleri modellerken ortaya çikan ve önceden anlatilan eksikliklerini gidermek üzere kabul görmüs bir metottur. RTP nesnelerin iç ve dis davranislarini birbirinden ayirmak için dogrudan ve genel bir yol sunar. Üstelik RTP, NYP kavramlarini dogal ve zarif bir sekilde genisletir.

Evrimlesen varliklari modellerken nesne düzeyinde özellestirme yapilmasi, sinif düzeyinde özellestirmeye göre daha iyi bir yaklasimdir. Bu durumda bir gerçek dünya varligi, her biri yükümlü oldugu görevlerden birini icra eden bir rol nesnesi olmak üzere, birden fazla nesne ile temsil edilecektir. RTP'de nesneler yeni roller alarak evrimlesir. Örnek düzeyinde gerçeklesen bu özellesmeye "nesne düzeyinde kalitim" adi verilir. NYP'de bir varligi modellemek için birden fazla nesne kullanildiginda, tüm bu nesnelerin ayni varligi simgeledigi gerçegi programci ek önlemler almadigi takdirde kaybolur. Rol modeli programcidan bu gibi yükleri alir ve nesne düzeyinde kalitim için bir mekanizma sunar.

Nesne düzeyinde kalitim "bir parçasidir" iliskisini (Zendler, 1998) basariyla modellerken sinif düzeyinde kalitim zarif bir biçimde "aynisidir" iliskisini (Zendler, 1998) modeller. Gerçek dünya sistemlerini modellerken her iki iliskinin kullanilmasi da gerekli oldugundan, bir nesne yönelimli ortamda iki tip kalitim da bir arada kullanilabilmelidir. Bu nedenle bir çok basarili rol modeli mevcut bir NYP dilini genisletecek sekilde gerçeklenmistir.

3 JAVA IÇIN BIR ROL MODELI: JAWIRO

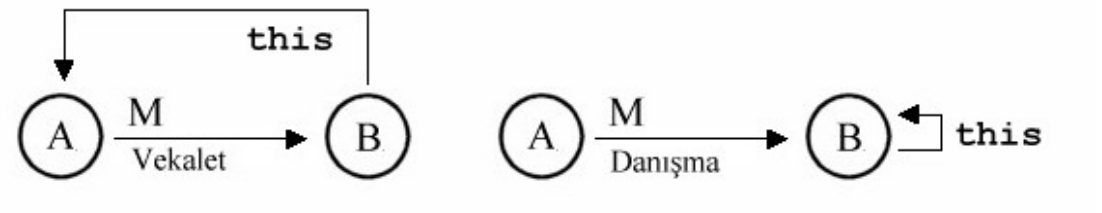
Java diline rol destegi kazandirmak üzere hazirlanan rol modeline JAWIRO (Java with Roles) adi verilmistir. JAWIRO rollerin kazanilmasina, terk edilmesine, geçici olarak askiya alinip sonra yeniden oynanilmasina ve alt rolleri ile birlikte baska bir sahibe aktarilmasina izin vermektedir. Bu bölümde gerçeklenen rol modeli ayrintili olarak incelenecektir.

3.1 JAWIRO Rol Modeli

JAWIRO rollerin tüm temel özelliklerini, herhangi bir ek kisitlama olmaksizin desteklemektedir. Tek kisitlama Java dilinin kendisinden gelen sinif düzeyinde çoklu kalitimin kullanilamamasidir. Ortaya çikan model sinif düzeyinde kalitim ile nesne düzeyinde kalitimin birlikte kullanilmasina izin veren, esnek ve basarima ek yük getirmeyen bir rol modelidir.

Rollerin iliskisel hiyerarsisini modellemek için agaç gösterimi kullanilmistir. Incelenen çalismalarin bazilarinda kullanilan küme gösterimiyle karsilastirildiginda, agaç gösterimi rol sahipligi iliskilerinin daha anlamli ve etkili olarak modellenmesini saglar. Küme seklinde gösterim diger bazi tasarim ayrintilari ile birleserek birçok çalismada rollerin temel özelliklerinden biri veya birkaçinin desteklenememesine neden olmustur.

JAWIRO nesne düzeyinde çoklu kalitimi destekler: Bir gerçek dünya sisteminin daha iyi modellenmesi için gerekli olan durumlarda, bir rol nesnesi farkli siniflara ait nesneler tarafindan sahiplenilebilir. Bir rol örnegi ayni anda yalniz bir sahip tarafindan oynanilabilecegi için bu durum herhangi bir belirsizlige sebep olmaz. Ancak JAWIRO'nun mevcut halinde nesne düzeyinde çoklu kalitim kullanilacagi zaman, programci rolün olasi sahiplerinin siniflarini ortak bir arayüz ile bir araya getirmelidir. Bu zorunluluk sadece alt rolün, üst rollerden birinin bir üyesine veya metoduna erismesi gerektigi durumlarda ortaya çikar. Yine de sik bir yaklasim olmayan bu zorunluluk, 5. bölümde anlatilan gelistirmelerden biri ile giderilecektir. JAWIRO, "this" isaretçisinin alinan mesajin iletildigi nesneyi gösterdigi "danisma" mekanizmasini kullanir. Bunun alternatifi olan vekalet mekanizmasinda ise "this" isaretçisi mesajin orijinal alicisini gösterir. Anlatilan iki mekanizma Sekil 1'de görülebilir.



Sekil 1 Vekalet ve danisma mekanizmalari.

JAWIRO, Kristensen tarafindan tanimlanan hali ile rollerin asagida listelenen tüm temel özelliklerini (Kristensen, 1996) gerçekler :

- *Görülebilirlik:* Bir nesneye erisim mevcut bakis açisi (rolü) ile sinirlidir.
- *Bagimlilik* : Roller sadece bir sahibe bagli olduklarinda anlamlidir.
- *Kimlik* : Bir gerçek dünya varliginin sahip oldugu rollerin tümü ile tanimlandigi gerçegi korunur. Bir baska deyisle her rol sahibini ve hiyerarsinin kökünü bilir.
- *Dinamizm* : Bir varlik hayati boyunca çesitli roller kazanabilir ve terk edebilir.
- *Çogulluk* : Bir varlik ayni rol tipinin birden fazla örnegine sahip olabilir. Nitelikli roller olarak adlandirilan bu tür rollerde ayni siniftan örnekler birbirlerinden bir niteleyici ile ayrilir.
- *Soyutlama* : Roller çesitli hiyerarsik iliskileri gerçekleyecek sekilde organize edilebilir.
- *Rollerin rolü* : Bir rol baska rollerin sahibi olabilir.

JAWIRO ek olarak su özelliklere de sahiptir :

- Sinif düzeyinde kalitim, nesne düzeyinde kalitim ile birlikte kullanilabilir.
- Bazi roller ortak davranis ve yapiya sahip olabilir. Bu özellik bir önceki sinif düzeyinde kalitim özelligi sayesinde mümkün olmaktadir.
- Nesne düzeyinde çoklu kalitim desteklenmektedir.
- Roller bir süreligine askiya alinabilir ve tekrar etkinlestirilebilir.
- Bir rol baska bir sahibe alt rollerini kaybetmeden aktarilabilir.
- Degisik roller çakismaya neden olmadan ayni adli üye ve metotlara sahip olabilir.
- Varliklar herhangi bir tip rolü oynayip oynamadiklari hakkinda sorgulanabilir.

3.2 Kullanici Arayüzü

JAWIRO'nun kullanici arayüzü ve bazi önemli üyeleri Tablo 1'de verilmistir. Bir rol hiyerarsisinin kökü olabilecek gerçek dünya nesneleri "Actor" sinifindan türetilir. Rol nesneleri ise "Role" sinifi ile modellenir. Nitelikli rollerin modellenmesinde kullanilan "AggregateRole" sinifi, "Role" sinifindan kalitim yolu ile türetilmistir. Hem rol hem de aktör nesnelerinin benzer davranislari olabilecegi için, "Actor" ve "Role" siniflari "RoleInterface" adli ortak bir arayüzü gerçekler.

Bir "Actor" veya "Role" nesnesine rol ekleme, rol geçisi, rol terki ve rol aktarimi komutlari, ilgili metotlarin açik anlasilir ad ve islevleri sayesinde kolayca kullanilir. Ancak rol varligini kontrol eden komutlarin açiklanmasi yerinde olacaktir.

Aykiri durumlardan kaçinmak için, "as" komutu ile rol geçisi yapmadan önce "canSwitch" metodu ile geçilmek istenen türde rolün elimizdeki nesnenin hiyerarsisinde o an bulunup bulunmadigini kontrol etmek dogru bir davranis olur. Bu açidan "as" ve "canSwitch" komutlari birbirini tamamlar. Rollerin herhangi bir anda terk edilebilecegi, askiya alinabilecegi veya bir baska sahibe aktarilabilecegi unutulmamalidir. Kalicilik destegi olan sistemlerde rol varligi kontrolü yapilmasi çok daha fazla önem tasir; Herhangi bir anda son durumu diskten yüklenen herhangi bir hiyerarside hangi rollerin olup hangilerinin olmadiginin derleyici ile kontrolü çok zor olacaktir. Ancak her seferinde "canSwitch" metodu ile kontrol yapmanin basarimi azaltacagi unutulmamali ve gereksiz kontrollerden kaçinilmalidir. Örnegin bir rol döngü içinde çalistirilacaksa, gerekli kontrol döngü blogu öncesinde yapilabilir.

Rol varliginin kontrolü için kullanilabilecek ikinci metot ise "canDelegate" metodudur. Bu komutla belli bir rol nesnesinin komutun çalistirildigi nesne ile ayni hiyerarside olup olmadigina bakilir. "canSwitch" metodunun aksine, "canDelegate" metodundan sonra "as" ile rol geçisi yapilmasina gerek yoktur. Örnegin a rol nesnesinden b rol nesnesine geçis yapilacaksa, if(a."canDelegate"(b)) sinamasi basarili ise b nesnesi dogrudan kullanilabilir.

Tablo 1 JAWIRO Kullanici Arayüzü

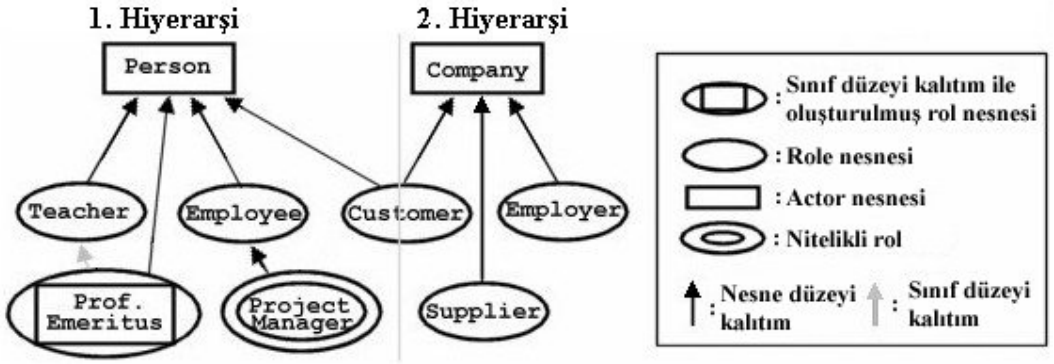
RoleInterface Arayüzünün Metotlari	
Metot Imzasi	**Islevi**
public boolean addRole(Role r)	Bu nesneye yeni bir rol ekler.
public boolean canDelegate(Role r)	r nesnesinin bu nesne ile ayni hiyerarside bulunup bulunmadigini arastirir.
public boolean canSwitch(String className)	Bu nesnenin aranan tipte bir role sahip olup olmadigini tüm hiyerarside arastirir.
public Object as(String className)	Rol geçisi komutu.
public Object as(String className, String identifier)	Nitelikli roller için rol geçisi komutu.
public boolean canSwitch(String className, String identifier)	Bu nesnenin aranan tipte bir nitelikli role sahip olup olmadigini tüm hiyerarside arastirir.
Actor Sinifinin Ek Üyesi	
Adi	**Islevi**
RoleHierarchy hierarchy	Bu "Actor" nesnesinin kökü oldugu hiyerarsiyi yönetir.
Role Sinifinin Ek Metotlari	
Metot Imzasi	**Islevi**
public Object playedBy( )	Bu rolü oynayan nesneyi döndürür.
public Object Actor()	Bu nesnenin rol hiyerarsisinin kökünü döndürür.
public boolean resign( )	Bu rolün kalici olarak terk edilmesi.
public boolean suspend( )	Bu rolün geçici olarak askiya alinmasi.
public boolean resume( )	Askiya alinan bu role devam edilmesi.
public boolean transfer(RoleInterface newOwner)	Bu rolün alt rolleri ile birlikte "newOwner" ile gösterilen yeni sahibe aktarilmasi.
Role Sinifinin Ek Üyeleri	
Adi	**Islevi**
RoleInterface owner	Bu rolü oynayan nesneyi tutar. Bu bir "Actor", "Role", veya "AggregateRole" örnegi olabilir.
Actor root	Rol hiyerarsisinin kökünü tutar.
AggregateRole sinifi için ek üye	
Adi	**Amaci**
String identifier	Nitelikli rollerin ayirt edilmesi için kullanilir.

3.3 JAWIRO ile Rol Kullanimi

JAWIRO ile rol kullaniminin örneklenmesi ve yeteneklerinin gösterilmesi için Sekil 2'de gösterilen rol sistemi kullanilmistir. Örnek hiyerarsi Sekil 3'te bir kismi verilen kod tarafindan sekilde görüldügü gibi kullanilabilir. Sekil 3'deki kodun tamami (Selçuk, 2004) adresinde bulunabilir.

Örnek hiyerarsiyi kullanan ve (Selçuk, 2004) adresinde bulunan kodlardan bazi kisimlarin açiklanmasi, JAWIRO ile rol kullaniminin daha iyi anlasilmasini saglayacaktir :

- Sekil 2'deki rol sisteminde iki ayri "Actor" nesnesinin olusturdugu iki ayri hiyerarsi mevcuttur. "Customer" rolünün belli bir örnegi ayni anda yalnizca bir hiyerarsiye ait olabilir, çünkü ayni anda yalniz bir "Actor" örnegi tarafindan oynanilabilir.
- "ProfEmeritus" sinifi, "Teacher" sinifindan sinif düzeyinde kalitim ile türetilmistir. Bu sekilde iki tür kalitimin bir arada kullanimi örneklenmistir.



Sekil 2 Iki hiyerarsiden olusan bir rol sistemi örnegi

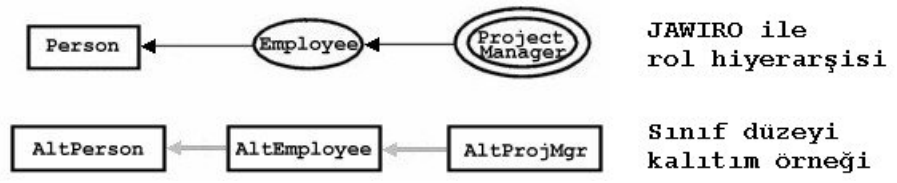
```
Company co1,co2; Employer er;
Supplier su; Customer cu1,cu2;
Person p1,p2; Teacher t;
ProfEmeritus pe; Employee ee1,ee2;
ProjectManager pm1,pm2;

co2=new Company("Black Mesa Labs","BML");
su=new Supplier(); co2.addRole(su);/*MTCX,
BML'nin kurumsal müsterisi oldu*/

co1=new Company("Metacortex","MTCX");
er=new Employer();
co1.addRole(er); //MTCX eleman alimina hazir
cu1=new Customer(su);
co1.addRole(cu1);
//MTCX, BML'nin müsterisi oldu

if(co1.canSwitch("examples.Customer"))
//Rol varligi kontrolü (1. yol)
  ((Customer)er.as("examples.Customer"))
  .buy(3); //MTCX buys 4 goods
//Rol geçisi: Isveren'den Müsteri'ye

p1=new Person("Tom Anderson","843-663");
t=new Teacher("Fizik");
p1.addRole(t); //Tom fizik ögretmeni oldu,
t.suspend();//ögretmenlige ara verdi ...
t.resume(); //ve görevine geri döndü.
ee2=new Employee(er,"453-543");
p2=new Person("Gordon Freeman","637-252");
p2.addRole(ee2); //Gordon, MTCX'e girdi

t.resign(); //Tom emekli oldu ama ...
pe=new ProfEmeritus(t);
p1.addRole(pe); /*...emekli profesör olarak
ders vermeye devam ediyor*/

ee1=new Employee(er,"628-749");
p1.addRole(ee1); //Tom da MTCX'e girdi.
pm1=new ProjectManager("Sanal G.","SG");
ee1.addRole(pm1);
//Tom sanal gerçeklik projesi yöneticisi oldu
pm2=new ProjectManager("Yapay Zeka","YZ");
ee1.addRole(pm2);
//Tom ikinci projeyi de yönetiyor.
pm2.transfer(ee2);//Görevi Gordon'a devretti.

cu2=new Customer(su);
p1.addRole(cu2);
//Tom, BML'nin bireysel müsterisi oldu.
if(p1.canDelegate(cu2))
//Rol varligi kontrolü (2. yol)
  cu2.buy(2); /*Dogrudan kullanim: Rol geçisi
  gereksiz,p1'in cu2'yi oynadigini ögrendik*/
```

Sekil 3 Örnek hiyerarsiyi kullanan kod parçasi

- "Person" ve "Company" siniflari, iki nedenle "BuyerAndSeller" arayüzünü gerçekler :
    i. Hisse senedi bilgisinin "Person" ve "Company" siniflarinin "stock" adli üyesinde tutulmasi seçilmistir.
    ii. "Customer" rolünü adi geçen iki sinifin örneklerinin de oynayabilmesi. Böylece nesne düzeyinde çoklu kalitim örneklenmis olur.
- "Employee" sinifi, "Employer" sinifindan "employer" adli bir üyeye sahiptir. "Employee" sinifindan bir nesne ile bunun "employer" üyesi farkli hiyerarsilerde bulunmaktadir. Buna ragmen "Employee" örnegi "employer" üyesi sayesinde iliskide bulundugu "Company" örneginin adini söyle ögrenebilir : "((Company) employer.Actor()).name"

3.4 Basarim Degerlendirmesi

Önerilen modelin uygulanabilirligini göstermek üzere, JAWIRO ile olusturulan bir rol hiyerarsisinin ve sinif düzeyinde kalitim ile elde edilen esdeger gerçeklemenin metot çalistirma süreleri ölçülmüstür. Örnek olarak Sekil 2'deki "Person" köklü hiyerarsinin "ProjectManager" nitelikli rolünü içeren dali seçilmistir. Bu dal ve esdegeri olan sinif düzeyinde kalitim hiyerarsisi Sekil 4'te gösterilmistir. "ProjectManager" ve "AltProjMgr" örneklerinin Sekil 5'te verilen "whoami" metodu çalistirilarak yapilan karsilastirmanin sonuçlari ise Tablo 2'de görülebilir.



Sekil 4 Basarim ölçümünde kullanilan hiyerarsiler

```
public void whoami( ) {
        System.out.print("My name is "+((Person)Actor()).name+" and ");
        System.out.println("I lead the "+projectName+" project." );
}
```

Sekil 5 Çalistirilan metot

"Example02.java" ... "Example06.java" adlari verilen ölçüm kodlarinin tamami (Selçuk, 2004) adresinde bulunabilir. Örnek nesnelerin "whoami" metodu 1000 kez çagirildiginda geçen süre "System.currentTimeMillis" metodu ile ölçülmüs ve sürelerin aritmetik ortalamasi alinmistir. Metot çagrilmadan önce ilgili hiyerarsinin de kurulmasi gerekmektedir. Hiyerarsinin kurulumu ile metot çagrimi ayri ayri ölçülmüs ve 10 ardisik oturumda elde edilen zamanlar milisaniye cinsinden Tablo 2'de verilmistir. Sonuçlar 256MB RAM ve 1.6GHz islemciye sahip, JDK 1.4.2 ve Windows 2000 isletim sistemi kurulu bir PC ile elde edilmistir.

Tablo 2 JAWIRO ile sinif düzeyi kalitimin basarimlarinin karsilastirilmasi

	Sinif Düzeyi Kalitim Hiyerarsisi (Example02.java)			**JAWIRO ile Rol Kullanimi (Example03.java)**		
Deneme	**Kurulum**	**Çalistirma**	**Toplam**	**Kurulum**	**Çalistirma**	**Toplam**
1	15	453	468	31	453	484
2	16	437	453	32	468	500
3	15	454	469	47	438	485
4	16	469	485	47	453	500
5	16	437	453	47	453	500
6	16	453	469	47	453	500
7	31	453	484	46	469	515
8	16	469	485	46	454	500
9	16	469	485	47	469	516
10	16	453	469	47	453	500
Ortalama	**17,3**	**454,7**	**472**	**43,7**	**456,3**	**500**

Sonuçlar incelendiginde rol hiyerarsisinin olusturulmasinin sinif düzeyinde kalitima göre %52 daha yavas olmasina ragmen JAWIRO ile metot çalismasinin sadece %0,35 daha yavas oldugu görülecektir. Metot çalismasinin hiyerarsi olusturmadan daha baskin bir kriter oldugu unutulmamalidir, çünkü tipik bir sistemde hiyerarsi sadece bir kez kurulduktan sonra sürekli metot çalistirmasi yapilacaktir. Hiyerarsilerin sik sik yeniden kurulacagi ortamlarda dahi toplam sonuçlara bakildiginda JAWIRO ile sinif düzeyinde kalitimin yaklasik ayni basarima sahip oldugu görülmektedir. Bununla beraber rol hiyerarsilerinin dinamik sistemleri daha iyi modelledigi ve sinif düzeyinde kalitimin getirdigi ek kesisim siniflari olusturulmasi geregini ortadan kaldirdigi unutulmamalidir.

(Selçuk, 2004) adresinde verilen "Example03.java" kodu incelendiginde, Tablo 2'de JAWIRO ile elde edilen sonuçlara rol varligi kontrolü ve ek rol geçisi komutunun dahil olmadigi görülür. "Example03.java" dosyasinda komutun 1000 kez "for" döngüsü içerisinde çalistirildigi kisim degistirilerek "Example04.java", "Example05.java", ve "Example06.java" siniflari elde edilmistir. Sadece degisen kisimlar Sekil 7, Sekil 8 ve Sekil 9'da verilmistir. Karsilastirma kolayligi için "Example03.java" sinifindaki döngü ise Sekil 6'da verilmistir. Bu üç sinifin önceden anlatilan test yöntemi ile bulunan basarim sonuçlari Tablo 3'te görülebilir.

JAWIRO'nun çesitli kullanimlarinin aldigi sürelerin sinif düzeyinde kalitim ile geçen süre toplu olarak karsilastirilmasi Tablo 3'te görülebilir. Tablo 4'te verilen rakamlar ise, JAWIRO kullanan örneklerin ortalama çalisma sürelerinin sinif düzeyinde kalitima göre yüzdelik olarak ne kadar fazla oldugunu gösterir.

```
for( i = 0; i < 1000; i++ ) {
      pm1.whoami();
}
```

Sekil 6 "Example03.java" içerisinde ölçülen döngü

```
for( i = 0; i < 1000; i++ ) {
      ((ProjectManager)p1.as("examples.ProjectManager")).whoami();
}
```

Sekil 7 "Example04.java" içerisinde ölçülen döngü

```
for( i = 0; i < 1000; i++ ) {
      if( p1.canSwitch("examples.ProjectManager", "VR"))
      ((ProjectManager)p1.as("examples.ProjectManager")).whoami();
}
```

Sekil 8 "Example05.java" içerisinde ölçülen döngü

```
for( i = 0; i < 1000; i++ ) {
      if( p1.canDelegate(pm1) )
            pm1.whoami();
}
```

Sekil 9 "Example06.java" içerisinde ölçülen döngü

Tablo 3 Rol geçisi ve/veya kontrolü eklendiginde elde edilen ölçüm sonuçlari

	Example04.java			**Example05.java**			**Example06.java**		
	Kurul.	Çalis.	Toplam	Kurul.	Çalis.	Toplam	Kurul.	Çalis.	Toplam
1	47	453	500	47	469	516	47	453	500
2	47	453	500	47	453	500	47	469	516
3	47	468	515	47	453	500	47	468	515
4	47	469	516	47	437	484	47	453	500
5	47	469	516	47	468	515	47	438	485
6	47	453	500	47	469	516	47	453	500
7	47	453	500	47	468	515	47	469	516
8	47	469	516	47	453	500	47	437	484
9	47	453	500	47	453	500	47	453	500
10	47	453	500	47	453	500	46	469	515
Ort.	**47**	**459,3**	**506,3**	**47**	**457,6**	**504,6**	**46,9**	**456,2**	**503,1**

Tablo 4'de görülebilecegi gibi, JAWIRO'nun rollerin dogrudan çalistirilmasinda (Sekil 6) sinif düzeyinde kalitimla denkligi rol geçisi ve kontrolü sirasinda da sürmektedir. "Example04" kodundaki sadece rol geçisinin ek yükünün "Example05" kodundaki rol kontrollü ve geçisli ek yükten daha küçük olmasi beklenirken aksinin çikmasinin nedeni, her iki kodun denkligi ve test yapildigi anda isletim sistemindeki anlik dalgalanmalardir. Baska zamanlarda yapilan testlerde son üç örnegin çalistirma ek yüklerinin %0,3 ile %1 arasinda ve ayni sirada kalmadan degistigi görülmüstür.

Tablo 4 Rol kullanimi örneklerinin sinif düzeyinde kalitima göre getirdigi ek yükler.

Örnek	Sinif düzeyinde kalitim ile arasindaki yüzde fark			Açiklama
	Kurulum	Çalistirma	Toplam	
Example02	0	0	0	Sinif düzeyinde kalitim
Example03	152,60	0,35	5,93	Sadece rol çalistirma
Example04	171,68	1,01	7,27	Rol geçisi ile çalistirma
Example05	171,68	0,64	6,91	Rol kontrollü ve geçisli çalistirma
Example06	171,10	0,33	6,59	Rol kontrollü çalistirma

Rollerin dogrudan çalistirilmasi ile varlik kontrolü yapildiktan sonra çalistirilmasinin denk zaman almasinin nedeni JAWIRO'da yapilan bir iyilestirmedir. Tüm rol geçisi ve rol varligi kontrolü komutlarinin gerektirdigi hiyerarside rekürsif arama isleminin "RoleHierarchy" sinifi tarafindan gerçeklestirildigi Tablo 1'de belirtilmisti. Geçis veya kontrol amaçli arama yapildiginda, en son bulunan rol nesnesi "lastHitObj" adli üyede saklanir ve sonraki çagri ayni rol nesnesine ise tekrar arama yapilmadan bu nesne gönderilir. Tutarliligin korunmasi için "transfer", "suspend", "resume" ve "resign" komutlari "lastHitObj" üyesini geçersiz kilar. Bu iyilestirme yapilmadan önce "Example04", "Example05" ve "Example06" kodlari "Example02" ve "Example03" kodlarina göre %3,5 kadar ek yük getirmekteydi.

4 ILGILI ÇALISMALAR

Java disindaki dillere dayanan rol modeli çalismalarinin incelenmesi ve karsilastirilmasi (Selçuk, 2003) makalesinde bulunabilir. Bu bölümde Java dilindeki güncel çalismalar olan Lee ve Bae'nin rol modeli (Lee, 2002) ve Dec-Java (Bettini, 2003) JAWIRO ile karsilastirilmistir. Yakin tarihli bir rol modeli olan ve C++ diline dayanan INADA (Aritsugi, 2000) ile Smalltalk diline dayanan güçlü bir rol modeli olan Gottlob'un çalismasi (Gottlob, 1996) eklenmistir.

INADA (Aritsugi, 2000) her tipin bir role denk geldigi kalici bir ortamda C++ diline rol destegi kazandirir. Roller arasindaki iliskiler, agaç gösteriminden daha zayif bir yapi olan küme seklinde modellenmistir. INADA'nin eksiklikleri nitelikli rolleri desteklememesi ve çalisma aninda tip kontrolü için bir yol sunmamasidir.

Gottlob tarafindan önerilen rol modeli (Gottlob, 1996) çok esnek bir yapiya sahiptir ve rollerin tüm temel gereksinimlerini bir kisitlama disinda destekler. Söz konusu kisitlama, nitelikli rollerin alt rollerinin de nitelikli rol olmasi zorunlulugudur. JAWIRO'da böyle bir kisitlama bulunmamaktadir.

Dec-Java (Bettini, 2003), donatici tasarim dokusundan (Gamma, 1994) esinlenerek hazirlanmistir. Dec-Java'nin rol modeli iki tip nesneye dayanir: Bir durumu temsil eden "bilesen" nesnesi "donatici" nesnelerin içine gömülmüstür. Donatici nesnesi, bir gerçek dünya nesnesinin anlik durumunu gerçek dünya nesnesinin davranisina yansitir. Bir bilesen nesnesi bir veya daha fazla donatici nesnesi tarafindan donatilabilir. Iç içe donatimlar da desteklenir, bu durumda en distaki donatici hem en içteki bilesene hem de en içteki donaticiya erisebilir. Dec-Java'nin bu özelligi rollerin iliskisel hiyerarsisini destekleyen bir RBP dili olarak kullanilabilmesine izin verir. Dec-Java rollerin tüm temel gereksinimlerini desteklemekle birlikte, sadece nesne düzeyinde kalitima izin vererek sinif düzeyinde kalitimi desteklemez. Ayrica Dec-Java mevcut haliyle geriye bir deger döndüren metotlari desteklemez ve metotlar sadece sonlarina donaticilar sayesinde kod eklenmesi ile evrimlestirilebilir.

Lee ve Bae (Lee, 2002) odak noktasi dinamik evrimlesme yerine modellenen sistemin yapisal kurallarinin çignenmesini ve anormal rol iliskilendirmelerinin önlenmesi olan bir rol modeli önermislerdir. Diger rol modellerinin aksine, gerçek dünya nesnelerini modelleyen "çekirdek" nesneler rol nesnelerine atanir. Bir çekirdek nesne birden fazla role sahip olacagi zaman, bireysel rol nesneleri tek bir büyük ve birlesik rol altinda birlestirilirler. Bu birlestirme, rollerin hiyerarsi halinde modellenmesini engelleyip küme seklinde gösterimi zorunlu kilar. Ayrica Lee ve Bae'nin rol modeli nitelikli rollerin desteklenmesini olanaksiz kilan bir yapida tasarlanmistir. Lee ve Bae'nin rol modelinin son eksigi ise "Select" seçim kuralinin henüz gerçeklenmemis olmasidir: Bir bilesik rolü olusturan rollerden ikisi arasinda ad çakismasi oldugunda, ki bu rollerin temel özelliklerindendir, "Select" seçim kurali çakisan ögelerden çalisma anindaki gereksinimlere göre dogru olan ögenin seçilerek kullanilabilmesini saglar.

5 DEGERLENDIRME VE GELECEKTEKI ÇALISMALAR

Java dilinin yansitma yeteneklerinin kullanimi en az düzeyde tutuldugu için JAWIRO metot çalistirma basarimina ek yük getirmemektedir. Üstelik bu sayede üçüncü parti araçlarin sundugu ek yansitma islevlerine ihtiyaç duymamaktadir. JAWIRO ile yetenekli bir rol sistemi gerçeklenmis olup, desteklenen islevlerde hiçbir kisitlama bulunmamaktadir. JAWIRO'nun önceki bölümde incelenen çalismalarla karsilastirmasi Tablo 5'te özetlenmistir.

Gelecekte yapilacak çalismalardan Iki, JAWIRO'ya kalicilik desteginin kazandirilmasidir. Böylece kullanicilar bütün bir rol hiyerarsisini diske kaydedip daha sonra yeniden kullanabileceklerdir. JAWIRO için planlanan ikinci gelistirme ise rol hiyerarsisindeki herhangi bir rolün bir üyesinin veya metodunun, ait oldugu üyeyi belirtmeksizin kullanilabilmesine izin verilmesidir. Ad çakismalari oldugu takdirde en çok evrimlesen rolün üyesine erisilecektir. Bu islev ayni zamanda nesne düzeyinde çoklu kalitim kullanildiginda üst rollerin siniflarinin ortak bir arayüzü gerçeklemesi zorunlulugunu ortadan kaldiracaktir. Bu zorunlulugun sadece alt rolün, üst rollerden birinin bir üyesine veya metoduna erismesi gerektigi durumlarda ortaya çiktigi hatirlatilir. Ancak söz konusu yetenek Java diline özgü yansitma yetenekleri ile gerçeklenmek zorunda oldugu için, rollerin dogrudan çalistirilmasina göre daha zayif bir basarim sergileyecektir.

JAWIRO'ya kazandirilmasi planlanmamis tek yetenek, modellenen gerçek dünya sisteminin belirledigi yapisal ve islevsel kisitlamalarin çignenmesini önlemektir. Yazarlar bu görevin rol modeline degil, programcilara ait oldugunu ve dikkatli bir tasarim ile denenmis yazilim mühendisligi metotlarinin kullanilarak kurallara uygun bir model kurulabilecegini düsünmektedir.

	(Gottlob, 1996)	**INADA (Aritsugi, 2000)**	**Dec–Java (Bettini, 2003)**	**(Lee, 2002)**	**JAWIRO**
Temel dil	Smalltalk	C++	Java	Java	Java
3. parti yazilima dayanmamasi	+	–	+	–	+
Nitelikli roller	+	–	+	–	+
Normal rollerin nitelikli rollerce oynanabilmesi	–	X	X	X	+
Hiyerarsi destegi	+	–	+	–	+
Sinif düzeyinde çoklu kalitim	+	+	X	X	X
Nesne düzeyinde çoklu kalitim	–	–	–	–	+
Sinif ve nesne düzeyinde kalitimin birlikte kullanimi	+	+	–	+	+
Çalisma aninda rol kontrolü	+	–	–	–	+
Kalicilik	–	+	–	–	–
Rol iliskilendirme anormalliklerinin önlenmesi	–	–	–	+	–
Ilgili rolü/rolleri bilmeden üye/metot erisimi	+	–	X	–	–

Tablo 5 JAWIRO'nun önemli rol modelleri ile karsilastirilmasi. Isaretlendirme: +: Destekleniyor, –: Desteklenmiyor, X: Uygulanamaz anlamindadir.

Kaynakça

(Aritsugi, 2000) M. Aritsugi, A. Makinouchi, "Multiple-Type Objects in an Enhanced C++ Persistent Programming Language", Software - Practice and Experience, Cilt 30, Sayi 2, Sayfa 151–174, 2000

(Bettini, 2003) L. Bettini ve digerleri, "Extending Java to Dynamic Object Behaviours", Electronic Notes in Theoretical Computer Science, Cilt 82, Sayi 8, 2003

(Drossopoulou, 2002) S. Drossopoulou ve digerleri, “More dynamic object reclassification : Fickle”, ACM Trans. Programming Languages and Systems. Cilt 2, Sayfa 153–191, 2002

(Gamma, 1994) E. Gamma ve digerleri, Design Patterns Elements of Reusable Object Oriented Software, ISBN: 0-201-63361-2, Addison Wesley, 1994.

(Gottlob, 1996) G. Gottlob ve digerleri, “Extending object-oriented systems with roles”, ACM Trans. on Information Systems, Cilt 14, Sayi 3, Sayfa 268–296, 1996

(Kristensen, 1996) B. B. Kristensen, “Conceptual Abstraction Theory and Practical Language Issues”, Theory and Practice of Object Systems, Cilt 2, Sayi 3, 1996

(Lee, 2002) J-S. Lee, D-H. Bae, “An Enhanced Role Model For Alleviating the Role-Binding Anomaly”, Software - Practice and Experience, Cilt 32, Sayfa 1317–1344, 2002

(Selçuk, 2003) Y. E. Selçuk, N. Erdogan, “How to Solve the Inefficiencies of Object Oriented Programming : A Survey Biased on Role-Based Programming”, 7th World Multiconf. Systemics, Cybernetics and Informatics. Cilt 13, sayfa 160–165, 2003

(Selçuk, 2004) “JAWIRO kullanim örnekleri”, Y.E. Selçuk, Son giris tarihi: 7.7.2004, Web adresi: http://www.library.itu.edu.tr/~yeselcuk/bilisim04.html, 2004

(Ungar, 1987) D. Ungar ve digerleri, “Self: The power of simplicity”, Proc. ACM Conf. on Object Oriented Programming Systems, Languages and Applications, (OOPSLA ’87), Sayfa 212–242, 1987

(Wong, 1997) R. K. Wong, ve digerleri, “A Data Model and Semantics of Objects with Dynamic Roles”, IEEE Int’l Conf. On Data Engineering, Sayfa 402-411, 1997

(Zendler, 1998) A. M. Zendler, “Foundation of the Taxonomic Object System”, Information and Software Technology, Cilt 40, Sayfa 475-492, 1998

Özgeçmis

Yunus Emre Selçuk. 1976 dogumludur. Ankara Tinaztepe lisesini 1993 yilinda bitirmistir. 1997 yilinda Istanbul Üniversitesi Bilgisayar Bilimleri Mühendisligi bölümünden lisans derecesini almistir. 2000 yilinda Istanbul Teknik Üniversitesi, Bilgisayar Mühendisligi bölümünde yüksek lisans çalismasini tamamlamistir. Halen Istanbul Teknik Üniversitesi, Bilgisayar Mühendisligi bölümünde doktora ögrencisi olup, ayni zamanda ITÜ Mustafa Inan Kitapliginda sistem yöneticisi olarak çalismaktadir.

Nadia Erdogan. Lisans ve Yüksek Lisans çalismalarini Bogaziçi Üniversitesi Elektrik Müh. Ve Bilgisayar Bilimleri Bölümlerinde, Doktora çalismasini da Istanbul Teknik Üniversitesi Elektrik-Elekronik Fakültesi, Kontrol ve Bilgisayar Müh. Bölümünde tamamlamistir. Halen, ITÜ Bilgisayar Müh. Bölümünde ögretim üyesi olarak görev yapmaktadir.