itüdergisi/***d***
mühendislik
Cilt:6, Sayı:3, 27-34
Haziran 2007

Nesneye yönelik programlamanın rol modelleri ile genişletilmesi

Yunus Emre SELÇUK*, Nadia ERDOĞAN

İTÜ Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Programı, 34469, Ayazağa, İstanbul

Özet

Gerçek dünyada karşılaşılan sistemler durağan ve dinamik olmak üzere iki kümeye ayrılabilir. Durağan sistemlerin aksine, dinamik sistemlerde zamanla değişen ve evrimleşen varlıklar söz konusudur. Bu şekilde zamanla değişen nesneleri nesneye yönelik programlamanın (NYP) sınıf hiyerarşileri ile modellemek zor olacaktır. Bu zorluğun nedeni, dinamik yapıdaki gerçek dünya varlıklarının durağan yapıdaki sınıflar ile modellenmeye çalışılmasıdır. NYP durağan sınıf yapısına dinamizm katacak yeteneklere sahip olmasına rağmen dinamik sistemlerin modellenmesinde karşılaştığı güçlüklerden tam anlamıyla kurtulamaz. Rol modelleri; dinamik sistemlerin nesneye yönelik programlama ile modellenmesinde karşılaşılan güçlüklerin çözümü için önerilen yollar arasında kullanışlı olmaları, NYP kavramları ile iyi uyuşmaları ve problemin çözümü için dolaysız bir yol sunmaları nedeniyle dikkati çekmektedir. Nesne düzeyinde özelleştirmeye dayanan rol modelleri, sınıf düzeyinde özelleştirmeye dayanan nesneye yönelik programlamayı doğal bir biçimde genişletir. Bu nedenlerle dinamik sistemlerin saf NYP yaklaşımı ile sunulandan daha iyi modellenebilmesi gereksinimini karşılamak üzere, Java programlama diline rol desteği kazandıran bir rol modeli olan JAWIRO (Java with Roles) gerçeklenmiştir. Bu çalışma sırasında rollerin diğer çalışmalarda önerilen özellikleri arasına altı yeni özellik kazandırılmıştır. Bu makalede rollerin JAWIRO rol modeli ile nasıl kullanılabileceğinin anlatılmasına ek olarak, JAWIRO'nun güncel rol modelleri ile rollerin gerçeklenen özellikleri ve çalışma anı başarımı açılarından karşılaştırmaları verilmiştir. Sonuç olarak JAWIRO, sahip olduğu özgün yetenekler ve çalışma anı başarımına ek yük getirmemesi sayesinde dinamik sistemlerin modellenmesi için önemli bir gereç haline gelmiştir.

Anahtar Kelimeler: *Rol modelleri, rol tabanlı programlama, dinamik sistemlerin modellenmesi, nesneye yönelik programlama.*

*Yazışmaların yapılacağı yazar: Yunus Emre SELÇUK. selcukyu@itu.edu.tr; Tel: (212) 285 33 00.
Bu makale, birinci yazar tarafından İTÜ Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Programı'nda tamamlanmış olan "Nesneye yönelik programlamaya rol desteğinin kazandırılması" adlı doktora tezinden hazırlanmıştır. Makale metni 06.07.2006 tarihinde dergiye ulaşmış, 27.07.2006 tarihinde basım kararı alınmıştır. Makale ile ilgili tartışmalar 30.04.2008 tarihine kadar dergiye gönderilmelidir.

Extending object oriented programming with role models

Extended abstract

Real world systems can be classified either as static or dynamic. In static systems, entities to be modeled can be divided into distinct classes and they do not change their classes. Such systems can be modeled efficiently by using the object oriented programming (OOP) paradigm as the relationship between an object and its class is persistent, static and exclusive in OOP. On the contrary, dynamic systems contain entities that constantly change and evolve. Modeling such entities with the class structure of OOP is hard because objects of dynamic nature are required to be modeled by classes having static nature. Although OOP has features to add some scale of dynamism to the static nature of its classes, it cannot fully eliminate the obstacles it faces while modeling dynamic systems.

When modeling dynamically evolving entities, using instance level specialization will eliminate the obstacles faced by the class level specialization. In that case, a real world entity is represented by multiple role objects, each representing one of the responsibilities of the real world entity. A role of an entity can be defined as the set of properties which are important for an object to be able to behave in a certain way expected by a set of other objects. The programming paradigm which is based on using roles is called role based programming (RBP), while a role model is software which specifies a style of designing and implementing roles. Role models provide a mechanism for object level inheritance while preserving the fact that multiple objects are used to model a real world entity.

Role models receive much attention with their usefulness among the proposed ways of modeling dynamic systems as they match with the OOP paradigm well and represent a direct way for solving the problem at hand. Role models which are based on object level specialization naturally extends the OOP which is based on class level specialization. Therefore, we have extended the Java programming language with role support in order to solve the problem of modeling dynamic systems better than the ways available with the pure OOP paradigm. For that purpose, a role model named JAWIRO (Java with Roles) is implemented.

Different researchers define the features expected from the roles in slightly different ways. According to our experience, the basic features of a role model should contain the following:

- *Roles can be gained and abandoned dynamically and independently of each other.*
- *Roles can be organized in various hierarchical relationships. A role can play other roles, too.*
- *An entity can switch between its roles.*
- *A role can access member variables and methods of other roles.*
- *Class level inheritance can be used together with object level inheritance.*
- *Entities can be queried whether they are currently playing a certain type of role or a particular role object.*
- *An entity can have more than one instance of the same role type. Such roles are called aggregate roles and distinguished from each other with an identifier.*
- *Different roles are allowed to have member variables and methods with same names without conflicts.*

These basic features do not cover all that can be done with roles. JAWIRO implements the following extended features of roles as well, where the last six are the unique contributions of JAWIRO:

- *Abnormal role bindings are prevented.*
- *Persistence is supported.*
- *Roles can be suspended and then resumed.*
- *A role can be transferred to another owner without dropping its sub roles.*
- *Multiple object level inheritance is supported.*
- *Any public member variable or method of any participant of a role hierarchy can be accessed solely by its name, without a direct reference to its owner. In case of identical names, the most evolved member is returned.*
- *By setting dominant nodes in a role hierarchy, the previous rule can be overridden.*
- *Both consultation and delegation mechanisms are supported.*

This paper describes how these features of roles are used with JAWIRO and compares recent role models in terms of features and run-time performance. As a result, JAWIRO is surfaced as an important tool for modeling dynamic systems, thanks to its unique contributions to features of roles and its neglible overhead on runtime performance.

Keywords: *Role models, role based programming, modeling dynamic systems, object oriented programming.*

Giriş

Gerçek dünyada karşılaşılan sistemler durağan ve dinamik olmak üzere iki kümeye ayrılabilir. Durağan sistemlerde modellenecek varlıklar ayrık sınıflara bölünebilir ve sınıflarını asla değiştirmezler. Sınıf tabanlı NYP yaklaşımında da, bir sınıf ile o sınıftan türetilmiş bir nesne arasındaki ilişki kalıcı, değişmez ve dışlamalı bir yapıya sahiptir. NYP'nin temellerinden olan sınıf kavramı durağan bir yapıya sahip olduğundan, durağan sistemler NYP ile kolayca modellenebilir.

Durağan sistemlerin aksine, dinamik sistemlerde zamanla değişen ve evrimleşen varlıklar söz konusudur. Buna en güzel örnek insanlardır. Bir insan hayatı boyunca birbirleriyle örtüşen değişik roller alır: Öğrenciliğe başlayan şahıs aynı zamanda yarı zamanlı bir veya birkaç iş sahibi olabilir; aynı anda bir eş, evlat, ebeveyn olmak gibi çeşitli yükümlülükleri yerine getirir.

Bu şekilde zamanla değişen nesneleri NYP'nin sınıf hiyerarşileri ile modellemek zor olacaktır. Bu zorluğun nedeni, dinamik yapıdaki gerçek dünya varlıklarının durağan yapıdaki sınıflar ile modellenmeye çalışılmasıdır. NYP bu durağan yapıya dinamizm katacak çeşitli yeteneklerine rağmen, dinamik sistemlerin modellenmesinde karşılaştığı güçlüklerden tam olarak kurtulamaz. Varlıkların birbirinden farklı rolleri bağımsız olarak alabilmesi bu güçlükleri arttırır.

Dinamik olarak evrim geçiren varlıkları modellemek için nesne düzeyinde özelleştirme yapılması, sınıf düzeyinde özelleştirmenin dinamik sistemlerin modellenmesi sırasında karşılaştığı güçlükleri çözer (Gottlob vd., 1996). Bu durumda bir gerçek dünya varlığı, her biri yükümlü olduğu görevlerden birini yürüten bir rol nesnesi olmak üzere, birden fazla nesne ile temsil edilir. Bir nesnenin bir rolü, bu nesnenin diğer nesnelerin beklediği gibi davranabilmesi için gereken bir özellikler kümesi olarak tanımlanabilir (Kristensen, 1996). Rol kullanımı temeline dayanan programlama biçimine rol tabanlı programlama (RTP), rollerin tasarımı için bir tarz ve rollerin kullanımı için çeşitli işlevler sunan yazılımlara ise rol modeli adı verilir. Rol modelleri nesne düzeyinde özelleştirme için bir yol sunar ve bir gerçek dünya varlığını modellemek için birden fazla nesne kullanıldığı gerçeğini korur.

Dinamik sistemlerin etkili ve kolay bir biçimde modellenmesi için önerilen yollar arasında rol modelleri; kullanışlı olmaları, NYP kavramları ile iyi uyuşmaları ve problemin çözümü için dolaysız bir yol sunmaları nedeniyle dikkati çekmektedir (Lee ve Bae, 2002). Nesne düzeyi özelleştirmeye dayanan rol modelleri, sınıf düzeyi özelleştirmeye dayanan NYP'yi doğal bir biçimde genişletir. Bu nedenlerle dinamik sistemlerin saf NYP yaklaşımı ile sunulandan daha iyi modellenmesi gereksinimini karşılamak üzere, Java programlama diline rol desteği kazandıran bir rol modeli olan JAWIRO (Java with Roles) gerçeklenmiştir.

Rollerin özellikleri

Kristensen (1996), Schrefl ve Thalhammer (2004), Selçuk ve Erdoğan (2004b) gibi değişik araştırmacılar, rollerin sahip olması gereken özellikleri birbirlerinden biraz farklı şekillerde tanımlamışlardır. Bu çalışma boyunca yapılan araştırmalar ve değerlendirmeler sonucunda, bir rol modelinin sağlaması gereken özellikler temel ve ileri düzey olmak üzere ikiye ayrılmıştır. Rollerin temel özellikleri şu şekilde sıralanabilir:

- Bakış açısı: Bir gerçek dünya varlığının oynadığı rollerin her biri, o varlığa bir bakış açısı oluşturur. Bir nesneye erişim o anda kullanılan rolü ile sınırlıdır. Ancak rol geçişi sayesinde, mevcut rol içerisinden başka bir rolün üyelerine erişim sağlanılabilir.
- Rol hiyerarşisi: Bir rol başka bir rolü oynayabilmelidir. Bu da rollerin çeşitli hiyerarşik düzenler oluşturacak şekilde ilişkilendirilebilmeleri anlamına gelecektir.
- Rol kazanımı ve terki: Roller birbirlerinden bağımsız olarak kazanılabilmeli ve terk edilebilmelidir.
- Birden fazla nesne ile gösterim: Bir gerçek dünya nesnesinin sahip olduğu rollerin tümü ile tanımlandığı gerçeği korunmalıdır. Her rol nesnesi sahibinden, kendi rollerinden ve hiyerarşi kökünden haberdar olmalıdır.

- Rol geçişi: Bir varlık herhangi bir rolüne herhangi bir anda geçiş yapabilmelidir.
- Sahibi üzerinden üye erişimi: Bir nesnenin bir rolü, diğer rollerinin üyelerine, üstteki iki özellikten biri sayesinde erişebilmelidir.
- Çakışma engeli: Değişik roller bir çakışma olmaksızın aynı adlı üye metot ve değişkenlere sahip olabilmelidir.
- Sınıf düzeyi kalıtım ile nesne düzeyi kalıtımın birlikte kullanımı: Nesne düzeyi kalıtım sınıf düzeyi kalıtımın yerini almaz. Aksine, nesne düzeyi kalıtım sınıf düzeyi kalıtımı tamamlar. Gerçek dünyada birlikte görülebilen bu iki tür kalıtım ilişkisi rol modellerinde de birlikte kullanılabilmelidir.
- Rol varlığı kontrolü: Varlıklar herhangi bir rol tipini oynayıp oynamadıkları hakkında sorgulanabilmelidir.
- Nitelikli roller: Bir varlık aynı rol tipinin birden fazla örneğine sahip olabilmelidir. Böyle rollere bu çalışmada nitelikli rol adı verilmiştir. Nitelikli roller birbirlerinden bir niteleyici ile ayrılırlar.

Rollerin temel özellikleri, rollerden beklenen yeteneklerin tümünü içermez. Rollerin aşağıda listelenen ileri düzey özelliklerinden ilk ikisi dışındakiler, bu çalışmanın özgün katkılarını oluşturur:

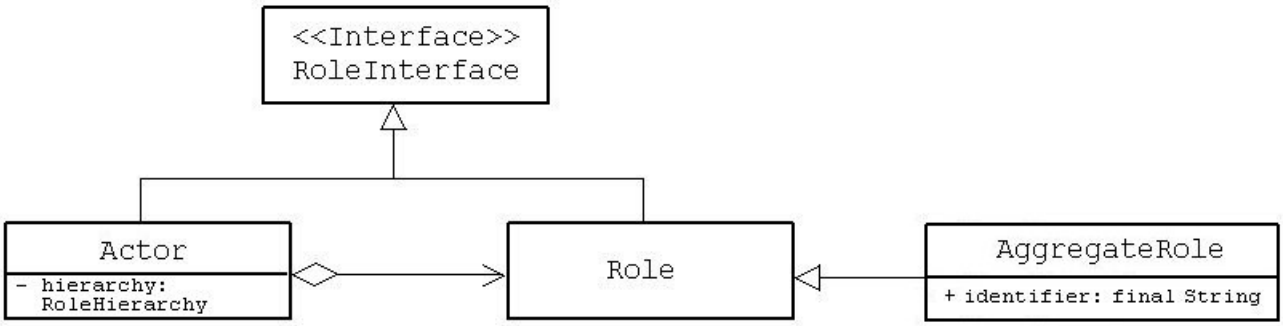
- Aykırı rol ilişkilerinin önlenmesi: Bir yazılımı hazırlayanlar ile kullananların farklı kişiler olmasından kaçınılamaz. Bu durumda kullanıcıların zamanla sistemin yapısına aykırı işler yapmaları olasılığı artacaktır. Anormal ilişkilerin önlenmesi bu nedenle bir zorunluluk olarak karşımıza çıkar.
- Kalıcılık: Kalıcılık yeteneğine sahip bir rol modeli bütün bir rol hiyerarşisini diskte saklayabilir ve sonradan tekrar kullanmak için diskten geri yükleyebilir.
- Danışma ve vekalet mekanizmalarının birlikte kullanımı: Rol geçişi sırasında mesajın ilk alıcısının saklanıp saklanmaması ile birbirinden ayrılan bu iki mekanizma gerçek dünya sistemlerinde bir arada görülebileceği için, bir rol modelinde de birlikte kullanılabilmelidirler.
- Rol aktarımı: Bir rol mevcut sahibinden bir başka sahibe, alt rollerini kaybetmeden aktarılabilmelidir.
- Nesne düzeyinde çoklu kalıtım: Bir rol örneği farklı sınıfların rol örnekleri tarafından oynanabilmelidir.
- Ad ile üye erişimi: Bir rol hiyerarşisinde bulunan nesnelerden herhangi birisinin istenilen bir üye değişkeni veya metoduna, sahibine doğrudan bir referans olmadan, sadece o üyenin adı belirtilerek erişilebilir. Aynı ada sahip birden fazla üye varsa, en çok özelleşmiş üyeye erişim sağlanmalıdır.
- Baskın roller: Önceki kural hiyerarşideki bazı katılımcıların baskın kılınması ile değiştirilebilmelidir.
- Rollerin askıya alınması ve askıdan indirilmesi: Geçici bir süre ihtiyaç duyulmayacak roller askıya alınabilmeli ve gerek duyulduğunda askıdan indirilerek durum bilgisinde kayıp olmadan tekrar kullanılabilmelidir.

JAWIRO rol modeli

Bu bölümde JAWIRO rol modeli hakkında genel bilgiler verilmiştir. Rollerin temel ve ileri düzey özelliklerinin JAWIRO ile kullanım örnekleri için Selçuk ve Erdoğan'ın (2004a) ve (2005b) çalışmaları incelenebilir.

Kullanıcı arayüzü

JAWIRO'nun kullanıcı arayüzünü ve bazı önemli üyelerini gösteren UML şeması Şekil 1'de gösterilmiştir. Bir rol hiyerarşisinin kökü olabilecek gerçek dünya nesneleri `Actor` sınıfından türetilir. Rol nesneleri ise `Role` sınıfı ile modellenir. Nitelikli rollerin modellenmesi kullanılan `AggregateRole` sınıfı, `Role` sınıfından kalıtım yolu ile türetilmiştir. Hem rol hem de aktör nesnelerinin benzer davranışları olabileceği için, `Actor` ve `Role` sınıfları, `RoleInterface` adlı ortak bir arayüzü gerçekler. Bu sınıfların kullanıcı arayüzünü oluşturan metotların imzaları ve kısa açıklamaları Selçuk ve Erdoğan'ın (2004a) ve (2004b) çalışmalarında bulunabilir. Kullanıcı arayüzünün bütün ayrıntılarına ise Selçuk'un (2005) web sitesinden erişilebilir.



Şekil 1. JAWIRO'nun kullanıcı arayüzü ve bazı önemli üyeleri

Nitelikli rollerin birbirlerinden ayırt edilmesini sağlayan niteleyici olarak bir karakter katarı olan `AggregateRole.identifier` üyesi kullanılmıştır. `AggregateRole` sınıfının bir varsayılan kurucu metodu yoktur; sahip olduğu tek kurucunun aldığı karakter katarı parametresi ise `identifier` üyesine değer atamak için kullanılır. Rol varlığı kontrolü ve rol geçişi komutlarının normal ve nitelikli rolleri destekleyecek biçimleri bulunmaktadır. Bu detayların dışında `Role` ve `AggregateRole` sınıfları arasında fark bulunmamaktadır.

`RoleHierarchy` sınıfı rol modelinin bel kemiğini oluşturur ve her `Actor` örneğinin bu türden hiyerarşi yöneticisi olarak adlandırılan bir üyesi bulunur. `RoleHierarchy` sınıfı kullanıcı arayüzünün bir parçası olmayıp metotlarına sadece `jawiro` paketinden erişilebilir. Hiyerarşi yöneticisi rol hiyerarşisini oluşturan ağacın düğümlerinde dolaşarak kullanıcı arayüzünü oluşturan metotların tüm gereksinimlerini karşılar.

JAWIRO rol modelinde rollerin ilişkisel hiyerarşisini modellemek için ağaç gösterimi kullanılmıştır. Ağaç gösterimi rol ilişkilerinin daha anlamlı ve iyi olarak modellenmesi ile rollerin özelliklerinin daha etkin bir biçimde gerçeklenebilmesini sağlar.

Kullanım örnekleri

JAWIRO ile rollerin temel ve ileri düzey özelliklerinin kullanımına dair örnekler için Selçuk ve Erdoğan'ın (2004a) ve (2004b) çalışmaları incelenebilir. JAWIRO ile rollerin özelliklerinin kullanımına örnek oluşturması amacıyla Şekil 2'deki kod parçası verilmiştir.

```
//Nesne tanımları
ActorPerson yunus, serdar, sena;
RoleDoctor docYunus, docSena;
RolePatient patSerdar;
/*Rol hiyerarşilerinin oluşturulma-
      sı.*/
//Köklerin oluşturulması.
yunus = new ActorPerson( "Yunus Em-
      re Selçuk", "212-7778899");
serdar = new ActorPerson( "Serdar
      Temiz", "216-9998877");
sena = new ActorPerson( "Sena Günay
      Selçuk", "212-7778899");
//Rollerin oluşturulması.
docYunus = new RoleDoctor("212-
      2853300");
docSena = new RoleDoctor("212-
      2853592");
patSerdar = new RolePatient("599-
      5556677");
//Rollerin eklenmesi
yunus.addRole(docYunus);
sena.addRole(docSena);
serdar.addRole(patSerdar);
//Rol varlığı kontrolü
if( yunus.canSwitch("RoleDoctor") )
  //Rol geçişi
  ((RoleDoctor)yunus.as
      ("RoleDoctor")).
      addPatient(serdar);
/*Doktor hastalanıyor ve görevi
başkasına aktarılıyor */
yunus.addRole( new
      RolePatient("598-9876543" );
docYunus.transfer(sena);
```

Şekil 2. Rollerin kullanımına örnekler

Şekil 2'de görülen `ActorPerson` sınıfının kurucusu kişinin adını ve ev telefonunu,

`RoleDoctor` sınıfının kurucusu ise doktorun iş telefonunu parametre olarak alır. `RoleDoctor` sınıfı doktorun tedavi ettiği hastaların listesini de tutar. Bu listeye yeni bir hasta eklemek için `addPatient` metodu kullanılır. `RolePatient` sınıfının ise iki kurucusu vardır. Bunlardan ilki sadece acil durumda aranacak telefon numarasını simgeleyen bir parametreye sahiptir. İkinci kurucu ise acil durum numarası ile birlikte hastanın atandığı doktorun adını da alır. Şekil 2'de geçen diğer komutlar ise JAWIRO rol modelinin kullanıcı arayüzüne aittir.

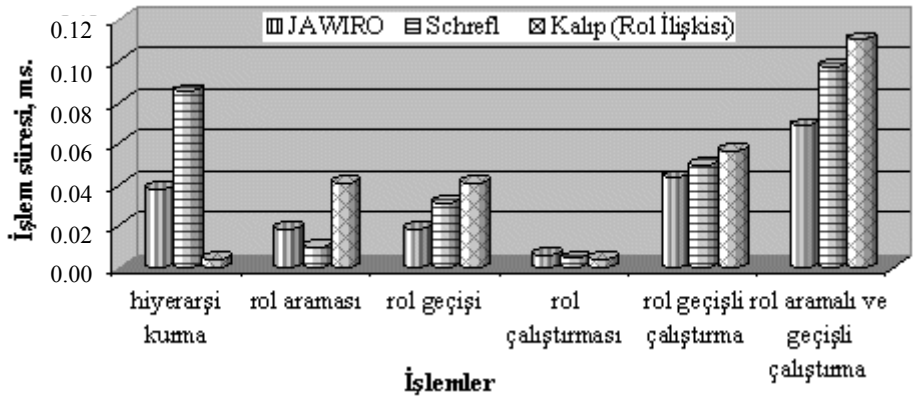
Şekil 2 sadece örnek olarak ele alınmalı ve bir kod parçası olarak düşünülmelidir. Şekil 2'deki rol varlığı kontrolünün ardından örnek kodda bir de rol geçişi yapılmasına gerek yoktur, çünkü programcının elinde doktor rolünü gösteren bir işaretçi zaten bulunmaktadır. Fakat gerçek dünya uygulamalarında çalışma anı sırasında her rol hiyerarşisinin tüm katılımcılarını gösterecek kadar işaretçinin tanımlanması pratik olmayacaktır. Örneğin bir hastane otomasyon yazılımında tüm kişilerin doktor ve hasta rollerini tek tek tutan değişkenler tanımlanması mümkün olmayıp, bu işaretçilerin bellekteki dizilerde veya tablolarda ayrı ayrı tutulması gereksiz olacaktır.

Güncel rol modelleri ile karşılaştırmalar

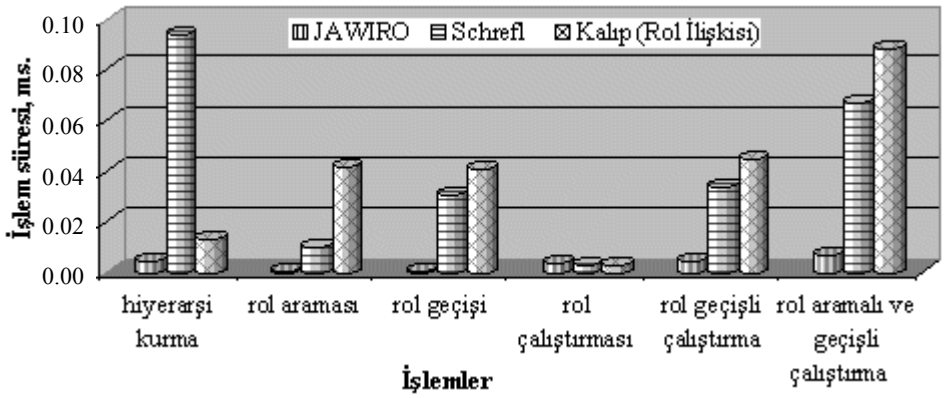
- *Rollerin desteklenen özelliklerine göre karşılaştırmalar* Bu bölümde JAWIRO öncelikle literatürdeki diğer rol modelleri ile rollerin gerçeklenen özellikleri açısından karşılaştırılacaktır. Özellik açısından yapılan karşılaştırmalara tasarım kalıplarının eklenmesi, başarılı tasarım kalıplarının güncelliğini yitirmeyerek yazılım uygulamalarına önemli katkılar sağlamaları nedeniyle yerinde olacaktır. Bu amaçla Fowler'ın çalışmasında incelediği tasarım kalıplarından rol ilişkileri tasarım kalıbı seçilmiştir. Bu tasarım kalıbının seçilme nedeni rollerin de rol sahibi olabilmesi dışında rollerin temel özelliklerinin tümünün desteklenmesidir. Roller ile ilgili diğer tasarım kalıplarında olduğu gibi, bu kalıp da rollerin hiyerarşik gösterimini destekleyecek şekilde genişletilebilir. Ancak bu kalıpların tümünde rol sahibi olabilecek sınıflar rol sınıflarına göre çok daha fazla karmaşıktırlar ve bu karmaşıklığı rol sınıflarına taşımak ise gerekli gerçekleme çabasını sınıf sayısı ile doğru oranda arttıracaktır. Tablo 1'de JAWIRO'nun seçilen çalışmalar ile özellik açısından karşılaştırılması görülebilir. Rollerin özelliklerine ek olarak, JAWIRO rol araması ve geçişinde çalışma anı başarımını arttırmaya yönelik bir iyileştirmeye sahiptir. Dolayısıyla bu iyileştirme sonraki alt bölümde incelenecektir.

Tablo 1. Güncel rol modellerinin rollerin desteklenen özellikleri açısından karşılaştırılması

	Jawiro	Schrefl (2004)	Rol İliş. Kalıbı	Dec-Java	Lee ve Bae	INADA
Temel Dil	Java	Java	Java	Java	Java	C++
Başka yazılıma bağlı olmama	+	+	+	+	+	–
Çakışma engeli	+	+	+	+	–	+
Çalışma anı rol varlığı kontrolü	+	+	+	–	–	+
Nitelikli roller	+	+	+	+	–	–
Rol hiyerarşileri	+	+	–	+	–	–
Aykırı rol ilişkilerinin önlenmesi	+	–	–	–	+	–
Nesne düzeyinde çoklu kalıtım	+	–	–	–	–	–
Ad ile üye erişimi	+	–	–	–	–	–
Baskın roller	+	–	–	–	–	–
Danışma ve vekaletin birlikte kullanımı	+	–	–	–	–	–
Rol aktarımı	+	–	–	–	–	–
Rollerin askıya alınması ve askıdan indirilmesi	+	–	–	–	–	–
Kalıcılık	+	–	–	–	–	+
Rol araması ve geçişinde iyileştirme	+	–	–	–	–	–



Şekil 3. Rol modellerinin rollerin temel özelliklerini çalıştırma süreleri



Şekil 4. İyileştirmeli çalışma düzeninde rollerin temel özelliklerini çalıştırma süreleri

- *Çalışma anı başarımına göre karşılaştırmalar* Çalışma zamanı başarımı da araçların değerini belirlemekte en az desteklenen özellikler kadar önemli bir ölçüttür: Kabul edilebilir bir sürede tamamlanamayan işlemler, uygulamalar için hiç bir yarar sağlamaz. Bu bölümde Schrefl'in (2004) hazırladığı siteden indirilebilen rol paketi, rol ilişkileri tasarım kalıbının bir gerçeklemesi ve JAWIRO rol modeli, çalışma anı başarımı açısından karşılaştırılmıştır. Bu amaçla rollerin temel özelliklerini simgeleyen komutlar bir çok kez çalıştırılarak, komutların tamamlama süreleri ölçülmüş ve aritmetik ortalamaları alınmıştır. Derecesi 3 ve derinliği 6 olan, tam dolu (363 rol nesnesine sahip) bir rol hiyerarşisinde yapılan karşılaştırmaların sonuçları Şekil 3'te verilmiştir. Test sisteminde 2.8GHz Pentium 4 işlemci, i865 yongalı anakart, 512MB RAM bellek ve Java 2 SE uyarlama 5.0.5 bulunmaktadır.

Uzun ömürlü sistemlerde bir varlığın çalışma anının herhangi bir noktasında sahip olacağı roller önceden derleme aşamasında bilinemez. Bu nedenle bir aykırı duruma yol açmamak için rol geçişi komutu vermeden önce, söz konusu varlığın istenen rolü oynayıp oynamadığını sınamak doğru olacaktır. Dolayısıyla bir rol sorgusunun ardından çoğu kez o role geçiş isteği gelecektir. Diğer rol modellerinin aksine JAWIRO sorgulanan son role ait bir işaretçi saklar. Takip eden geçiş isteği az önce aranan role aitse, rolün tekrar aranması yerine saklanan işaretçi sayesinde doğrudan rol geçişi yapılır. Ölçüm programı komutlarının JAWIRO'nun bu özgün katkısından yararlanılacak sırada verilmesi ile elde edilen sonuçlar ise Şekil 4'te görülebilir. Bu çalıştırma tarzına iyileştirmeli çalışma düzeni adı verilmiştir.

İyileştirmesiz çalışma düzeninde önce tüm rol nesneleri için rol varlığı kontrolü komutu verilmekte, bu komutların tamamlanmasının ardından tüm rol nesnelerine geçiş için komutlar ve-

rilmektedir. İyileştirmeli düzende ise her bir ayrı rol nesnesinin varlığının kontrolünün hemen ardından o rol nesnesine geçiş komutu verilmektedir. Şekil 3 ve Şekil 4 birlikte incelendiğinde JAWIRO'nun iyileştirmeli çalışma düzeninde başarımının büyük oranda iyileştiği görülebilir. Daha ayrıntılı başarım ölçümleri için ise Selçuk ve Erdoğan'ın çalışması (2005a) incelenebilir. Bu bölümde karşılaştırılan çalışmalar hakkında genel bilgi için ise Selçuk ve Erdoğan'ın bir başka çalışması (2004b) incelenebilir.

Sonuçlar

Dinamik sistemlerin etkin olarak modellenmesi için rol modelleri tek çözüm değildir. Ancak rol modelleri kolaylıkları, kullanışlılıkları ve NYP kavramları ile iyi uyuşmaları nedeniyle dikkati çekmektedir (Lee ve Bae, 2002). Bu çalışmada gerçeklenen rol modeli olan JAWIRO ise, sahip olduğu özgün yetenekler ve çalışma anı başarımına ek yük getirmemesi sayesinde dinamik sistemlerin modellenmesi için önemli bir gereç haline gelmiştir. JAWIRO rol modeline ve dokümantasyonuna Selçuk'un (2005) web sitesinden erişilebilir.

Kaynaklar

Betini, L., Capecchi, S. ve Venneri, B., (2003). Extending Java to dynamic object behaviours, *Electronic Notes in Theoretical Computer Science*, **82**.

Gottlob, G., Schrefl, M. ve Röck, B., (1996). Extending object-oriented systems with roles, *ACM Trans. on Information Systems,* **14,** 268-296.

Kristensen, B. B., (1996). Conceptual abstraction theory and practical language issues, *Theory and Practice of Object Systems,* **2,** 143-160.

Lee, J-S. ve Bae, D-H., (2002). An enhanced role model for alleviating the role-binding anomaly, *Software-Practice and Experience,* **32,** 1317-1344.

Schrefl, M. ve Thalhammer, T., (2004). Using roles in Java, *Software-Practice and Experience,* **34,** 449-464.

Selçuk, Y. E. ve Erdoğan, N., (2004a). Java diline ileri düzey rol desteği kazandırılması: JAWIRO, *Bildiri Kitabı,* Havacılıkta İleri Teknolojiler ve Uygulamaları Sempozyumu, Cilt II, 601-606, İstanbul.

Selçuk, Y. E. ve Erdoğan, N., (2004b). JAWIRO: An extended role model for Java, *Proceedings,* Int'l. Conf. on Computational Intelligence, 207-210, İstanbul.

Selçuk, Y. E. ve Erdoğan, N., (2005a). A role model for description of agent behavior and coordination, *Proceedings,* 6th Int'l Workshop on Engineering Societies in the Agents' World, 227-237, Kuşadası.

Selçuk, Y.E. ve Erdoğan, N., (2005b). Using roles with JAWIRO, Proceedings, AAAI 2005 Fall Symposium Series, Roles: An Interdisciplinary Perspective subtopic, Arlington, Virginia.

Fowler, M., (1997). Dealing with Roles. http://martinfowler.com/apsupp/roles.pdf, (13.12.2005)

Schrefl, M., (2004). The Java Role Package. http://www.dke.uni-linz.ac.at/research/projects/roles/index.html, (13.12.2005)

Selçuk, Y. E., (2005). JAWIRO rol modeli. http://www.yunusemreselcuk.com/jawiro/giris.html, (13.12.2005)