Hareketli Veriler İçin İş Varlığı Altyapısı (İVA)

Ozan Çetin

REHİS / EH-GYM / ASELSAN, Ankara, Türkiye
ozancetin@aselsan.com.tr

Özet. Kullanıcı arayüzünün esas teşkil ettiği uygulamalarda, verinin bağlamlar (context)/katmanlar arasında hareketi esnasında iş varlıklarının çoklanmalarına ve/veya uygulama seviyesinde veri bütünlüğünü sağlamak için birbirleri ile karşılaştırılmalarına ihtiyaç duyulabilir. Bu bildiride alana özel bir dille tanımlanan üst-veriler ile iş varlıklarına, kopyalama, karşılaştırma, sürümleme ve geri alma/tekrar uygulama yeteneklerini kazandıran İş Varlıkları Altyapısı'nın (İVA) kullanım durumları, mimari tasarımı, performans ölçümleri ve geliştirilebilecek yönleri ele alınmıştır.

Anahtar Kelimeler: alana özel dil, nesneye dayalı programlama, yazılım çerçeveleri

Abstract. In applications where the user interface is the main constituent, duplication of business entities while moving data between application contexts/layers or comparison of business entities between each other to preserve application level data integrity may be needed. In this work the use cases, architecture, performance measurements and rooms for improvement for the Business Entity Framework which provides copy, compare, versioning and undo/redo capabilities to business entities by means of the meta-data defined with a domain specific language is presented.

Keywords: domain specific language, object oriented programming, software frameworks

1 Giriş

Kullanıcı arayüzü yoğun uygulamalarda veri; bağlam/katman değişikliği, verinin merkezi olmaması gibi nedenlerle farklı mantıksal/fiziksel konumlar arasında hareket edebilir. Verinin hareketi esnasında iş varlıklarının [1], çoklanmalarına ve/veya uygulama seviyesinde veri bütünlüğünü sağlamak amacıyla birbirleriyle karşılaştırılmalarına ihtiyaç duyulabilir. Aynı zamanda yavaşça değişen boyutlar (slowly changing dimensions) gibi problemlerin çözülmesi, savunmaya dayalı kopyalar yaratılması (defensive copying) veya denetleme kayıtları tutulması için iş varlıklarının anlık görüntülerinin (snapshot) alınması da gerekebilir [2]. Genellikle tüm nesne hiyerarşisinin derin kopyasının (deep-copy) oluşturulması için bir formatta sıralama (serializa-

tion) yapılıp aynı formattan okuma (deserialization) yapılarak nesne tekrar oluşturulur fakat sıralama yöntemleri genellikle yavaştır. Derin-kopyaların oluşturulması veya nesnelerin karşılaştırılması için yansıtma (reflection) kullanan Kryo, Java Deep Cloning, veya Java Object Diff gibi kütüphaneler de bulunmaktadır [3] [4] [5]. Bunlar sıralama yöntemine göre daha performanslıdır fakat bu kütüphaneler nesne hiyerarşisini olduğu gibi kopyalar veya karşılaştırır, işlemlerde kullanılacak alanları basit bir biçimde seçmek mümkün değildir [6]. İVA, farklı fiziksel konumlarda, farklı uygulama katmanlarında ve/veya iş varlığı modeli içindeki farklı bağlamlarda, iş varlıklarının kullanımına yardımcı olmak amacıyla, iş varlıklarına kopyalama, karşılaştırma, sürümleme, geri alma/tekrar uygulama yeteneklerini kazandıran bir altyapıdır. İVA, iş varlığı modeline, alana özel bir dille (domain specific language), Java ek bilgileri (annotation) ve arayüzleri (interface) olarak eklenen üst verilerden (meta-data) ve bu üst verileri çalışma zamanında işleyerek yukarda bahsedilen yetenekleri sağlayan Java kütüphanelerinden oluşur. Bu bildiride Java ek bilgileri olarak eklenen alana özel dil ve bu dilin çalışma zamanında işlenip kullanılmasına yönelik tasarım, kullanım durumları, performans ölçümleri ve daha sonra altyapıya eklenmesi planlanan yetenekler anlatılacaktır.

2 Kullanım Durumları

İVA temel olarak bir nesnenin kopyasının çıkarılması istenen durumların hepsinde kullanılabilir. Bu durumlar aşağıdaki gibi özetlenebilir.

- Bağlam değişikliği: Verinin yeniden kullanılabilir olduğu her durumda bir bağlamda saklanan veri diğer bağlamlara aktarılabilir. Örneğin bir radar verisinin Radar Kütüphanesi'nden Plan'a, Plan'dan Senaryo'ya aktarımı bu kapsamda değerlendirilebilir.
- Katman değişikliği: Verinin birden fazla katmanda farklı kopyalarının bulunması gerektiği durumlarda kullanılabilir. Haberleşme katmanından asenkron olarak başka bir yazılıma gönderilecek nesne bilgilerinin iletilmeden önce kopyasının çıkarılması örnek gösterilebilir.
- Geri Alma / Tekrar Uygulama: Genellikle kullanıcı arayüzlerinde belirli aralıklarla veya kullanıcı etkileşimi ile kullanıcının üzerinde çalıştığı nesnelerin kopyaları yaratılabilir, daha sonra bu kopyalar üzerinde ileri geri gidilerek nesnenin o anki bilgilerine ulaşabilir. Örnek olarak nesne bilgileri güncellenirken pop-up açılan ekranlar öncesi, ekranda kullanılacak alt bilgilerin, daha sonra kullanıcı pop-up ekranda yaptığı değişiklikleri iptal etmek istediğinde, değişikliklerin geri alınacağı şekilde kopyalarının çıkarılması gösterilebilir.
- Denetleme Kayıtları Tutulması: Veri üzerinde yapılan değişikliklerin takibi amacıyla kopyalar oluşturulması gerekebilir. Denetleme kayıtları (audit trail) tutulması istenen nesne bilgileri için nesne kaydından "nesne denetleme" kaydına kopyalama yapılması bu kapsamda değerlendirilebilir.
- Sürümleme: Bir verinin değişik sürümlerinin tutulması amacıyla kopyalar oluşturulması gerekebilir.

– Anlık Kopya Oluşturma: Başka nesnelere referans verilip ileride nesne değiştiğinde bundan olumsuz etkilenilecek durumlarda referans yerine nesnenin kendisinin kopyası çıkarılarak referans yerine sahiplenme (composition) ilişkisi yaratılması gerekebilir.

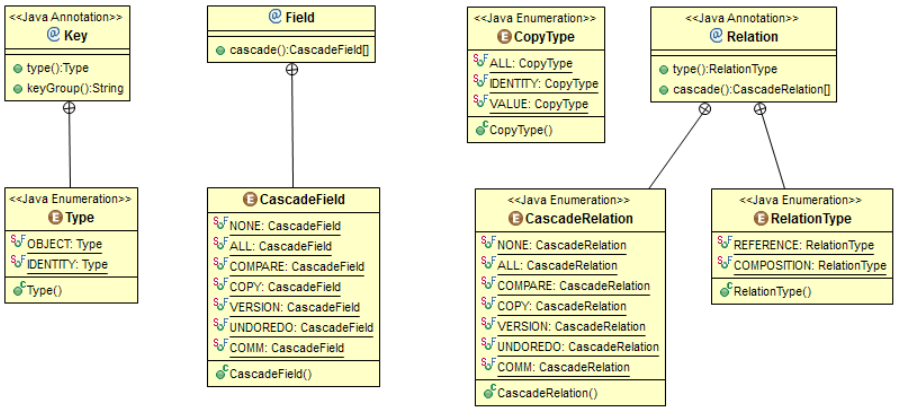
Bazı durumlarda daha önce kopyası çıkarılan nesnelerin birbirleri ile karşılaştırılması da gerekebilir. Bu duruma dair örnekler de aşağıda verilmiştir.

– Bağlamlar arası karşılaştırma: Bağlamlar arası aktarılan veriler bağlamlarda ayrı ayrı güncellenebilecekleri için birbirleri ile karşılaştırılıp yapılan değişiklikler gözlemlenebilir.
– Sürümler arası karşılaştırma: Bir nesnenin farklı sürümleri arasındaki değişiklikler takip edilmek istenebilir.
– Anlık kopya oluşturma: Anlık kopya oluşturulan durumlarda anlık kopyanın oluşturulduğu asıl nesne değişikliğe uğradığında, bazen değişikliklerin anlık kopyalara da aktarılması istenebilir, bunun için aktarım öncesi değişikliklerin de gözlemlenmesi gerekir.

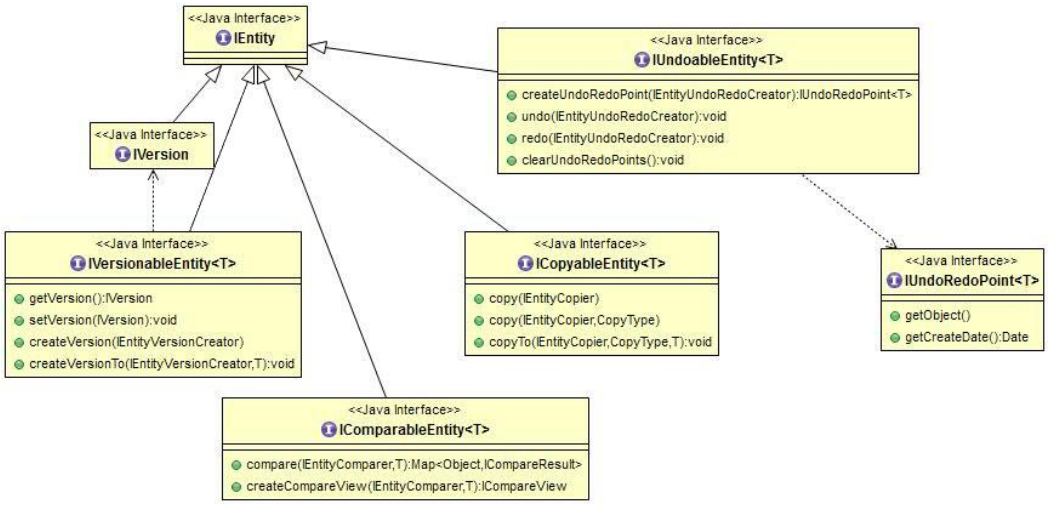
3 Alana Özel Dil ve Mimari

İVA temel olarak, nesnelerin özelliklerinin (property) ek-bilgiler ve arayüzler ile geliştirme aşamasında etiketlenip, daha sonra nesne hiyerarşisinin, çalışma zamanında yansıtma kullanımı ile özyinelemeli (recursive) olarak, etiketlerde belirtilen bu üst verilerin de yardımıyla, gezilmesi ve kopyalama/karşılaştırma işlemlerinin gerçekleştirilmesi prensibine dayanır.

Etiketleme esnasında kullanılan ek-bilgiler ve arayüzler Şekil 1 ve Şekil 2'de verilmiştir.



Şekil 1. Ek-bilgiler



Şekil 2. Arayüzler

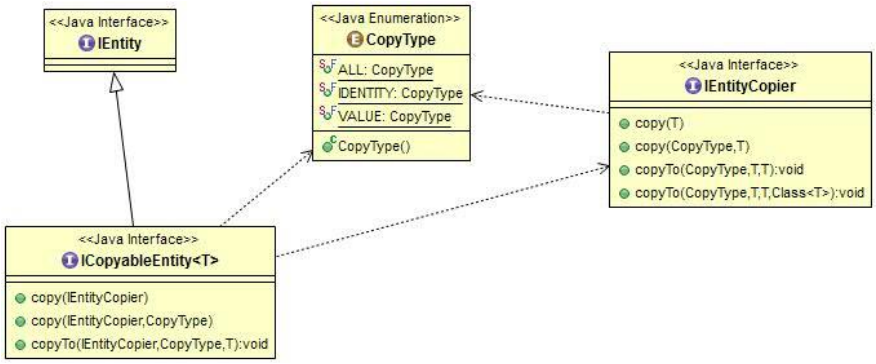
Üst verilerde kullanılan kavramlar aşağıda açıklanmıştır.

- Alan (Field): İşlemler sırasında bir nesnenin hangi alanlarının kopyalanacağı/karşılaştırılacağı bilgisi bu ek-bilgi ile sağlanır.
- İlişki (Relation): Karmaşık tipler arasındaki ilişkileri tanımlamak için kullanılır. İlişkinin işlemler sırasında göz önüne alınıp alınmayacağını, alınacak ise hangi tür ilişki olduğu bilgisini gösterir. İlişkiler referans (Reference) ve sahiplenme (Composition) tipinde olabilir. Referans alanlar kopyalanırken sadece adresleri kopyalanır, sahiplenilmiş alanlar kopyalanırken alanın kopyası öz-yinelemeli olarak çıkarılır.
- Anahtar(Key): Kopyalama/karşılaştırma sırasında hangi verilerin hangileriyle karşılaştırılacağının belirlenmesi veya kopyalar arasındaki mantıksal bağlantının sağlanması gibi problemleri çözmek için verileri tekil veya bir bağlam içinde tanımlayan anahtar alanlara ihtiyaç vardır [7]. Anahtar alanlar Nesne (Object) ve Kimlik (Identity) olmak üzere iki tiptedir. Kopyalama amacına göre bu alanlar korunur veya sıfırlanır. Örneğin sürümleme yapıldığında, kimlik anahtarları korunurken nesne anahtarları sıfırlanır. Ancak geri alma/tekrar uygulama amaçlı kopyalamalarda her iki türdeki anahtarlar da korunur. Karmaşık anahtarlar yaratmak için farklı anahtar alanlar tek anahtar grubu altında gruplanabilir.
- Kopya Tipi (CopyType): Kopyalama işleminde hangi anahtar alanların korunacağı bilgisini belirtir. Hepsi (All), Kimlik (Identity) ve Değer (Value) olmak üzere üç tipi vardır. Hepsi seçildiğinde nesne ve kimlik anahtarları ile alanların değerleri korunur, Kimlik seçildiğinde kimlik anahtarlarıyla alanların değerleri ve son olarak Değer seçildiğinde de sadece alanların değerleri korunur.

Altyapı ile kullanılması düşünülen sınıfların IEntity arayüzünü gerçeklemeleri gerekmektedir. Bazı üst-verilerin ek-bilgiler ile değil de ara yüzler ile sağlanmasının nedeni yardımcı olarak kullanılacak sınıflarda tip güvenliği (type-safety) sağlamaktır.

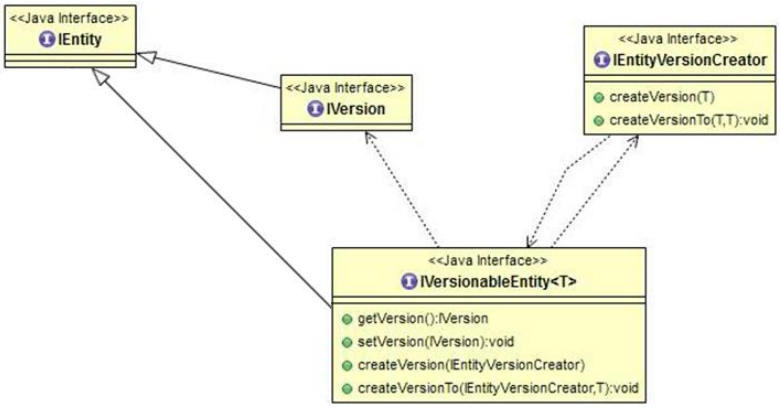
Temel yetenekler olan kopyalama, sürümleme, geri alma/tekrar uygulama ve karşılaştırma aşağıda sırası ile açıklanmıştır.

- Kopyalama: Kopyalanacak nesnenin (kaynak) ilkleyicisi çağrılarak yeni bir nesne yaratılır (hedef), daha sonra kaynak nesnenin kopyalanacak olarak etiketlenmiş tüm özellikleri derinlik-öncelikli (depth-first) şekilde gezilir. Eğer özellik basit (primitive) ise kopyalanarak hedef nesneye aktarılır. Eğer özellik karmaşık bir tipse, özelliğin üst veride yer alan tipine bakılır. Referans tipinde olan ilişkiler için özellik (yani obje referansı) olduğu gibi hedef nesneye aktarılır, sahiplenme tipinde olan ilişkiler için öz-yinelemeli olarak kopyalama işlemi tekrar çağrılır. Özelliğin çoklu nesne (Collection veya Array) tipinde olması durumunda listedeki her eleman için yukarda anlatılan yöntem ayrı ayrı uygulanır. Yaratılan tüm karmaşık tipler hızlı erişim sağlayan bir veri yapısı (hash-map) ile tutulur. Bunun nedeni karmaşık hiyerarşilerde bir nesneye birden fazla referans verilebilmesidir. Karmaşık bir nesne kopyalanmadan önce, bu veri yapısı üzerinden nesnenin daha önce yaratılıp yaratılmadığı kontrol edilir. Bu kontrol tüm durumları kapsamadığı için kopyalama işlemi tamamlandıktan sonra hiyerarşi ikinci bir kez gezilerek referans tipindeki ilişkiler tekrar kontrol edilir. Son olarak veri yapısı içinde yer alan kopyalanmış tüm nesnelerin anahtarları kopyalama tipine göre sıfırlanır. Kopyalamada kullanılan temel sınıflar ve arayüzler Şekil 3'te verilmiştir.



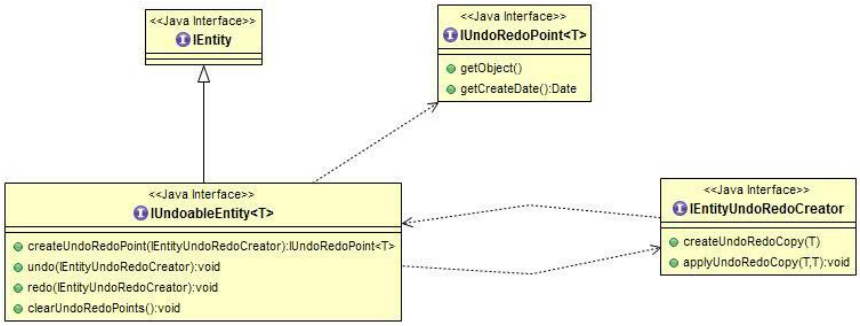
Şekil 3. Kopyalama Temel Arayüzleri

- Sürümleme: Temel yapı kopyalama ile aynı olmakla birlikte sürümlenen nesnenin sürümü gösteren bir alanı olması gerekir. Sürümlemede kullanılan temel sınıflar ve arayüzler Şekil 4'te verilmiştir.



Şekil 4. Sürümleme Temel Arayüzleri

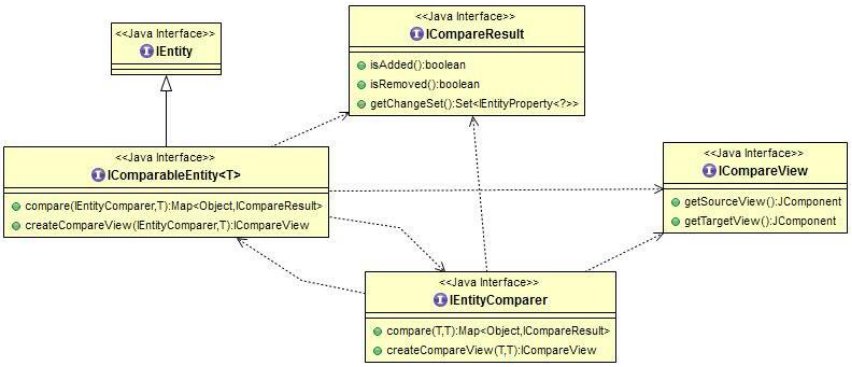
– Geri Alma / Tekrarlama: Geri alma / Tekrarlama'nın çalışma mantığında geri alınacak / tekrarlanacak anlık kopyalar (UndoRedoPoint) oluşturmak vardır. Fakat bu işlem yapılırken anlık kopyası oluşturulan nesnenin referansının değişmemesi gerektiğinden anlık kopyalar bir yığın (stack) yapısında tutularak geri alma/tekrarlama işleminde bu yığın yapısından alınan anlık kopyalardan asıl nesneye veriler aktarılır. Geri Alma / Tekrarlamada kullanılan temel sınıflar ve arayüzler Şekil 5'te verilmiştir.



Şekil 5. Geri Alma/Tekrarlama Temel Arayüzleri

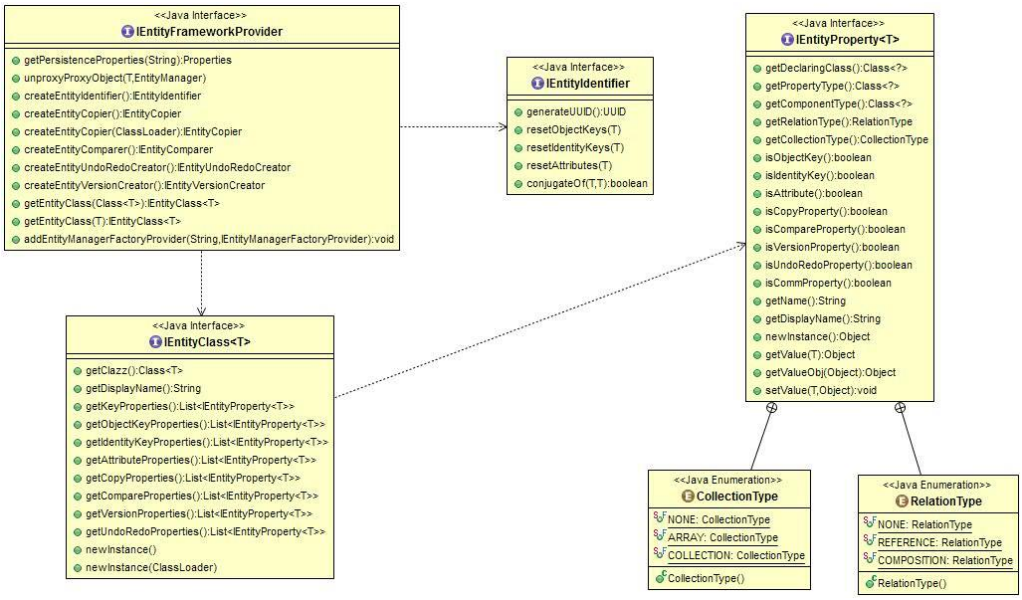
– Karşılaştırma: Karşılaştırılacak nesnelerden kaynak nesnenin karşılaştırılacak olarak etiketlenmiş tüm özellikleri derinlik-öncelikli (depth-first) şekilde gezilir. Primitif özellikler hedef nesne ile direk karşılaştırılır. Karşılaştırma sonucunun farklı olması durumunda alan değişmiş olarak işaretlenerek hızlı erişim sağlayacak bir veri yapısına atılır. Karmaşık nesneler için eğer ilişki tipi referans ise anahtar alanlar karşılaştırılır, ilişki tipinin sahiplenme olması durumunda öz-yinelemeli olarak karşılaştırma işlemi tekrar çağrılır. Özelliğin Collection veya Array tipinde olması durumunda alanlar primitif ise liste indekslerine göre liste elemanları karşılaştırılırlar. Liste elemanlarının karmaşık tipte olması durumunda ise önce listeler üzerinde

birbirleri ile karşılaştırılabilecek nesneler anahtar alanlar yardımı ile eşlenir, daha sonra birbirleri ile eşlenmiş nesneler için karşılaştırma işlemi öz-yinelemeli olarak çağrılır. Herhangi bir karşılığı bulunmayan nesneler silinmiş veya eklenmiş olarak işaretlenerek bir veri yapısında tutulur. Karşılaştırma işlemi sonrasında, değişmiş, eklenmiş ve çıkarılmış nesneler ile değişmiş alanlar, veri yapıları içinde belirlenmiştir. İstemciler istedikleri şekillerde bu farkları gösterebilirler, fakat nesneleri karşılıklı olarak ağaç yapısında değişmiş, eklenmiş, çıkarılmış alanları ile gösteren bir Swing bileşeni hazırlanmıştır. Karşılaştırmada kullanılan temel sınıflar ve arayüzler Şekil 6'da verilmiştir.



Şekil 6. Karşılaştırma Temel Arayüzleri

– Yardımcı fonksiyonlar: Bu işlemlerin altyapı tarafından hızlı bir şekilde yapılabilmesi için üzerinde işlem yapılacak tüm sınıflar kullanılmadan önce analiz edilir ve yardımcı fonksiyonlar bir önbellekte tutulur. Bu önbellek istemcilerin kullanımına da açıktır. Yardımcı fonksiyonlar için kullanılan temel sınıflar ve arayüzler Şekil 7'de verilmiştir.



Şekil 7. Yardımcı Fonksiyonlar

4 Performans Ölçümleri ve Karşılaştırma

Kopyalama performansı için derin-kopyalama yöntemleri araştırılmış ve seçilen kütüphaneler ve yöntemler karşılaştırılmıştır. Seçilen yöntemler ikili sıralama (binary serialization), XStream kütüphanesi ile XML formatında sıralama, Kryo kütüphanesi kullanımı ve Java Deep Cloning kütüphanesi kullanımıdır [8]. Testler Windows 7 SP1 yüklü, Intel i5-3470 işlemci ve 8GB ram bulunduran bir iş istasyonunda, Java 1.8.051 sürümü ile gerçekleştirilmiştir.

İlk önce nispeten basit, tek seviye nesne hiyerarşisi ve 10 adet basit alanı olan Sinyal nesnesi, seçilen yöntemler ile 100 defa kopyalanmıştır. Testler 5'er defa tekrar edilip ortalamaları alınmıştır. Sonuçlar Tablo 1'de verilmiştir.

Yöntem	**Süre (ms)**
İVA	232
İkili Sıralama	226
Java Deep Cloning Kütüphanesi	52
Kryo Kütüphanesi	61
XStream ile Sıralama	520

Tablo 1. - Sinyal Nesnesi 100 kopyası

Daha sonra daha karmaşık, 7 seviye nesne hiyerarşisi, her seviyede 20'den fazla alanı, çoklu nesneleri ve değişik seviyeleri arasında nesne referansları olan Radar

nesnesi her bir yöntem ile 100 kez kopyalanmıştır. Testler 5'er defa tekrar edilip ortalamaları alınmıştır. Sonuçlar Tablo 2'de verilmiştir.

Yöntem	**Süre (ms)**
İVA	3331
İkili Sıralama	3353
Java Deep Cloning Kütüphanesi	2337
Kryo Kütüphanesi	1158
XStream ile Sıralama	80909

Tablo 2. - Radar Nesnesi 100 kopyası

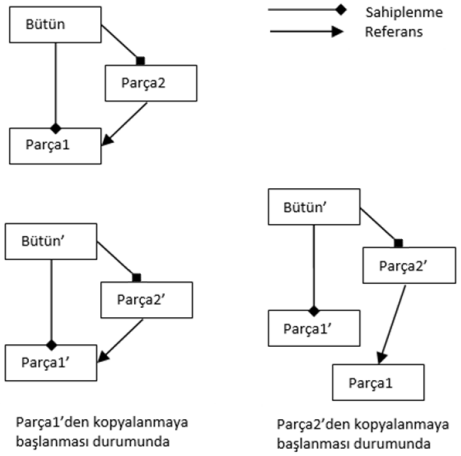
Bu sonuçlara göre İVA performans olarak ikili sıralama yöntemine yakındır. Cloner ve Kryo kütüphaneleri ile olan farkların nedenleri incelendiğinde aşağıdaki sonuçlara ulaşılmıştır.

Kryo ve Cloner kütüphaneleri alanlara erişim için "*java.lang.reflect.Field*" nesnesini kullanmaktadır. İVA alanlara erişim için alanların set ve get metotlarını "*java.lang.reflect.Method*" ile kullanmaktadır. İVA nesnelerinde, sonradan-yükleme (lazy-loading) desteği için, "*set*" ve "*get*" metotları ile çalışma zorunluluğu olduğu için bu kısımda bir değişiklik yapılamamaktadır. Fakat deneme amaçlı "*java.lang.reflect.Field*" kullanımına geçildiğinde Tablo 3'te verilen sonuçlarla karşılaşılmıştır.

Yöntem	**Süre (ms)**
İVA	3331
İVA (*Field* tabanlı)	2620
İkili Sıralama	3353
Java Deep Cloning Kütüphanesi	2337
Kryo Kütüphanesi	1158
XStream ile Sıralama	80909

Tablo 3. - Radar Nesnesi 100 kopya (*Field* tabanlı İVA ile)

Ayrıca Kryo ve Cloner kütüphaneleri derin-kopyalama mantığında karşılaştıkları tüm karmaşık nesneleri kopyaladıkları için nesne hiyerarşisini ikinci bir kez gezmeye ihtiyaç duymamaktadırlar. İVA karmaşık nesneleri ilişkilerin türüne göre seçimli olarak kopyaladığı için nesne hiyerarşisini ikinci bir kez gezmeye ihtiyaç duymaktadır. Şekil 8'de bu durumlardan birine örnek verilmiştir.



Şekil 8.

Bütün nesnesi kopyalanırken, Parça1'den kopyalanmaya başlanması durumunda hata oluşmamaktadır. Fakat Parça2'den kopyalanmaya başlanması durumunda Parça2'den referans verilen nesne orijinal Parça1 olarak kalmaktadır. Tüm ilişkilerin sahiplenme olarak kabul edildiği Kryo ve Cloner kütüphanelerinde kopyalama sırası önem arz etmemektedir.

İkinci gezinti mantığı İVA'dan kaldırıldığında Tablo 4'teki sonuçlarla karşılaşılmıştır.

Yöntem	**Süre (ms)**
İVA	3331
İVA (*Field* tabanlı)	2620
İVA (*Field* tabanlı + ikinci gezinti iptal)	1908
İkili Sıralama	3353
Java Deep Cloning Kütüphanesi	2337
Kryo Kütüphanesi	1158
XStream ile Sıralama	80909

Tablo 4. - Radar Nesnesi 100 kopya (*Field* tabanlı ve ikinci gezintisi iptal edilmiş İVA ile)

Görüldüğü gibi bu durumda performans diğer kütüphanelere yaklaşmaktadır.

5 Geliştirilecek Yönler / Eklenecek Yetenekler

Meta-verilerin ek-bilgiler ve arayüzler ile sağlanması hali hazırdaki POJO'ların (Plain Old Java Object) altyapı ile kullanımını engellemektedir. Arayüzlerle sağlanan üst-veriler ek-bilgilerle sağlanarak ve bu tip üst-verilerle ilgili kontroller tamamen

çıkarılarak arayüz bağımlılığı ortadan kaldırılabilir. Daha sonra ek-verilerin XML yapılandırma dosyaları yardımıyla da sağlanabilmesi için bir çalışma yapılabilir. Böylece herhangi bir POJO'nun da altyapı ile kullanımı gerçekleştirilebilir.

Karşılaştırma sırasında birebir aynı olmayan fakat küçük farkları olan alanları ihmal etmek için, karşılaştırılacak alan meta-verilerine "ihmal edilecek fark" verisi eklenebilir.

Karşılaştırılacak ve kopyalanacak alanlar için üst-veriler şu anda statik tanımlanmaktadır. İlerde üst-verilere birden fazla yapılandırma eklenerek veya yazılım arayüzüne üst-verileri çalışma zamanında ezecek şekilde araçlar koyularak daha dinamik bir kullanım sağlanabilir.

Hâlihazırda İVA ek-bilgileri ve ara yüzlerini destekleyen bir varlık ilişkileri modeli oluşturup ondan otomatik olarak kod üreten GMF tabanlı bir modelleme aracı bulunmaktadır. Fakat GMF tabanlı bu aracın üzerinde istenildiği şekilde değişiklik yapmak zaman alıcı ve maliyetli olmaktadır. Daha üst düzey bir kullanıcı deneyimi oluşturmak, dile yeni eklenecek özellikleri araca daha kolay entegre edebilmek için Sirius tabanlı yeni bir modelleme aracı geliştirilebilir [9].

İşlemler şu an için tüm koleksiyonlar için yapılamamaktadır. "*Map*" arayüzünden türeyen koleksiyonlar için destek eklenmesi planlanmaktadır.

İVA hali hazırda veri tabanı işlemlerini JPA (Java Persistence API) ek-bilgileri ile sağlamaktadır. JPA oldukça olgun bir API olduğu için veri-tabanı işlemleri ile ilgili dile eklenti yapılması düşünülmemektedir, fakat doğrulama (validation) ve kullanıcı arayüzüne bağlama (binding) ile ilgili meta-verilerin dile eklenmesi çalışmalarına devam edilmektedir. Böylece İVA ile nesneler veri tabanından kullanıcı arayüzüne kadar tüm katmanlarda kullanılabilecektir.

6 Sonuç

Hareketli verilerin bağlamlar/katmanlar arasında çoklanmalarına ve/veya uygulama seviyesinde veri bütünlüğünü sağlamak için birbirleri ile karşılaştırılmalarına ihtiyaç duyulmaktadır. Makalede İVA'nın hareketli verilerin değişik senaryolar için kopyalanma ve karşılaştırılma gereklerini nasıl karşıladığı, hali hazırda bulunan kütüphanelerden farkları ve bu kütüphanelere yönelik performans karşılaştırılmaları verilmiştir. İVA aktif olarak kullanıldığı uygulamaların değişik katmanlarında iş varlıklarının tekrar kullanımını sağlayıp, bazı sık kullanılan işlemleri otomatik hale getirdiğinden, yazılım geliştirmeye olumlu yönde katkı sağlamaktadır. İVA'nın geliştirilecek yönleri / eklenecek yetenekleri ile beraber yeni projeler kapsamında da kullanılması düşünülmektedir.

Kaynakça

1. Sutherland, J.: Why I love the OMG: emergence of a business object component architecture. StandardView, 6.1, 4-13 (1998)
2. Santos, V., Belo, O.: Slowly Changing dimensions Specification a Relational Algebra Approach. International Journal on Information Technology 1.3, 63-68 (2011)

3. Kougios, K.: Java Deep Cloning Library (Version 1.7.9) [Yazılım], https://github.com/kostaskougios/cloning/ (2016)
4. Grotzke, M., Sweet, N.: Kryo (Version 3.0.3) [Yazılım], https://github.com/EsotericSoftware/kryo (2016)
5. Bechler, D.: Java Object Diff (Version 0.93.2) [Yazılım], https://github.com/SQiShER/java-object-diff (2016)
6. Porres, I., Alanen, M.: A Generic Deep Copy algorithm for MOF-Based Models. Model Driven Architecture: Foundations and Applications. (2003)
7. Khoshafian, S. N., Copeland, G. P.: Object Identity. In: Proceedings of the 1986 ACM Conference on Object-Oriented Programming Systems, Languages and Applications, pp 406-416, ACM, New York (1986)
8. Walnes, J., Schaible, J., Talevi, M., Silveria, G.: XStream (Version 1.4.9) [Yazılım], http://x-stream.github.io/
9. Sirius [Yazılım], http://eclipse.org/sirius/