itüdergisi/***d***
mühendislik
Cilt:1, Sayi:2, 00-00
Aralik 2002

Kalici nesneleri destekleyen genisletilebilir bir sistem

Erdal KEMIKLI*, Nadia ERDOGAN

ITÜ Elektrik-Elektronik Fakültesi, Bilgisayar Mühendisligi Bölümü, 34469, Ayazaga, Istanbul

Özet

Bilgisayar sistemlerinin giderek daha önemli bir bölümünü olusturan yazilim bileseni program gelistirmede yasanan verimsizligin sorunlari ile yüzyüzedir. Bu makalede programci verimliligi problemini azaltacak genisletilebilir bir sistemi tanimlanmaktadir. Tasarlanan sistem degisik uygulama türlerini destekleyebilecek bir esneklige sahiptir. Kalici Nesneleri Destekleyen Genisletilebilir bir Sistem (KGS) genisletilebilir sistemler için kalici nesne kavramina dayanan üniform bir model önermektedir. KGS ile kullanicilar veri ve programlari kalici nesneler olarak degerlendirip tek bir arayüz ile yönetebilmektedir. KGS, nesneler arasi haberlesme hizmeti, istekçi nesne yöneticisi, aktif nesne kütüphanesi, adlandirma ve güvenlik sunucusu, nesne sunucusu ve bir önderleyici ile Linux isletim sistemi üzerinde gerçeklenmistir.
***Anahtar Kelimeler**: Kalici nesne, Isletim sistemi, genisletilebilir sistem, programlama.*

An extensible system supporting persistent objects

Abstract

Although software has become the most important component of computer systems, software production is suffering from a chronic crisis of unproductivity. Delayed and over budget software projects are very common. Many different approaches including the total quality management and integrated software development environments are considered in response to this problem. This paper defines an extensible and tailor able computing system model which will attack the programmer productivity issue from the technical side. The resulting system is suitable to be used as a base for an extensible system for different types of application domains. Extensible Persistent System (EPS) suggests a new model for extensible systems based on a unifying view of persistency. Data and processes are regarded as passive object and active object , and viewed as persistent objects, and handled through a uniform interface. EPS has extensive support for active object development. EPS also suggests new techniques for capability management, a ticket used to access resources on the system. EPS-C, a C programming language flavor is defined as part of the system. EPS is implemented by following components; inter-object communication service (IOC), client object library (COL), active object library (AOL), naming and protection server (NPS), object server (OBS), and EPS-C preprocessor on Linux operating system.
***Keywords:** Persistent object, operating system, extensible system, programming.*

*Yazismalarin yapilacagi yazar: Erdal Kemikli. erdalk@otokoc.com.tr; Tel: (212) 229 95 55 dahili: 1606.
Bu makale, birinci yazar tarafindan ITÜ Elektrik-Elektronik Fakültesi'nde tamamlanmis olan "Kalici nesneleri destekleyen genisletilebilir bir sistem" adli doktora tezinden hazirlanmistir. Makale metni 06.05.2002 tarihinde dergiye ulasmis, 13.09.2002 tarihinde basim karari alinmistir. Makale ile ilgili tartismalar 28.02.2003 tarihine kadar dergiye gönderilmelidir.

Giris

Klasik sistemlerde programlama dilleri geçici veriler için çok iyi bir destek saglarlar. Daha uzun ömürlü veriler ise VTYS (Veri Tabani Yönetim Sistemi) veya bir dosya sistemi araciligi ile yönetilir. Kendisini yaratan programdan daha uzun ömürlü verileri "kalici veri" olarak tanimliyoruz. Kalicilik kavrami bir verinin kendisine olan gereksinim devam ettigi sürece varligini sürdürebilmesini içermektedir. Bunun sonucunda, kalici sistemler veri yönetimi için tek bir soyutlama saglayabilirler. Bu soyutlama verinin farkli bellek sistemlerine aktarimi sirasinda gerek duyulan dönüsüm ihtiyacini ortadan kaldirarak yazilimcilari önemli miktardaki gelistirme çalismasindan kurtarir. PS-Algol'un tasarimcilari tarafindan yapilan bir arastirmada programlarin %30'unun veri yükleme/saklama/dönüsüm islemlerinden olustugu görülmüstür (Cockshot vd., 1984, Atkinson vd., 1983). Kalici nesneler için destek genellikle PS-Algol'da oldugu gibi programlama dili düzeyinde saglanmistir. Kalici nesne desteginin programlama dili düzeyinde verilmesinin iki sakincasi vardir (Kemikli ve Erdogan, 1997); isletim sistemleri uygulama için gerekli destegi vermeyebilir ve çabalar her yeni kalici nesne destekleyen dil için tekrar harcanacaktir (Kemikli ve Erdogan, 1998). Bu nedenler baska bir yaklasimi tesvik etmektedir; kalici nesne desteginin isletim sistem düzeyinde saglanmasi.

Isletim sistemleri genel kullanima uygun olma ve özel konularda uzmanlasma arasinda bir denge kurmak zorunda kalmaktadir. Genel amaçli bir sistem biraz çaba ile çok degisik uygulamalarda kullanilabilirken, uzmanlasmis sistemler belirli tip uygulamalari çok daha sorunsuz olarak çalistirabilmektedir. Genisletilebilir bir sistem, bir uygulamanin gereksinimleri dogrultusunda degistirilebilir bir sistemdir (Bershad vd., 1995).

Bu makalenin amaci, yazilim verimliligi konusuna teknik açidan bir çözüm öneren, kalici nesneleri destekleyen genisletilebilir bir sistem mimarisini açiklamaktir. Bu model pasif ve aktif nesne olarak adlandirilan iki temel soyutlamadan olusmaktadir.

- Pasif nesne, kalici özelligi olan veridir.
- Aktif nesne ise kalici özelligi olan veri ile onlari isleyebilen metodlardan olusur ve sistem davranisini genisleten bir sunucu olarak çalisir.

Çalismada nesne erisimi ve senkronizasyonu gibi konularda çoklu erisim politikalari ve dagitik hareketler gibi yeni yöntemler denenmistir.

Kalici Nesneleri Destekleyen Genisletilebilir Sistem (KGS) (Kemikli ve Erdogan, 1999) aktif/pasif nesne soyutlamasina dayali modellemesi sayesinde yazilimcilarin yeni uygulama ve sistemleri daha hizli bir sekilde gelistirmesine yardimci olur ve kalici nesneler ve aktif nesne gelistirme araçlari ile programcilarin isini kolaylastirir. Bunun yanisira, yeni sistem ek sistem yönetimi ve konfigürasyon yönetimi gibi çalismalar gerektirmez (Kemikli ve Erdogan, 2002). Sistem soyutlamasina yeni bir yaklasim getiren yenilikci bir yazilim gelistirme ve sistem modelleme araci olmasina ragmen, KGS çok yaygin bir ortam olan UNIX'e benzedigi için ögrenilmesi kolay bir sistemdir.

KGS'te sistemin basit ve varolan sistemlere benzer olmasi, diger taraftan da yeni bir programlama modeli ve araçlarla gelistirilmesi amaçlanmistir. Ayrica kolay ve anlasilir basit soyutlamalar sayesinde, gelistirilmesi kolay ve çesitli araçlar ve seçenekler ile kontrolun yazilimcida oldugu bir ortam tanimlanmistir.

Sekil 1. KGS bilesenleri

KGS'nin modelini gelistirme asamasinda çesitli alanlardaki arastirmalardan edinilen deneyimlerden yararlanilmistir. Çalismaya iliskin önemli arastirmalarin yapildigi alanlardan birisi isletim sitemleridir. Konuyla iliskili baska bir arastirma alani ise programlama dilleri ve soyutlama teknikleridir. Kalici veriler konusundaki arastirmalar ise en önemli arastirma alani olarak degerlendirilmektedir. KGS modeli tanimi degisik katmanlardaki çesitli bilesenlerden olusmaktadir (Sekil 1). En üst katmanda C dilinin gelistirilmis bir versiyonu olan yeni bir programlama dili KGS-C yer alir. Yeni programlama dilinin varolan bir dilin gelistirilmesiyle olusturulmasi sayesinde ögrenme süresinin azalacagi düsünülmektedir. KGS-C ile yazilan programlar KGS-C önislemcisi yardimi ile derlenirler. Baglanma asamasinda, görev düzeyinde islemlerden sorumlu sistemin karmasikligini iyi tanimlanmis ilkellerden olusan bir arayüz ile yazilimcidan soyutlayan bir kütüphane (Istemci Nesne Yöneticisi- INY) programa baglanir. Yazilimcilara temel aktif nesne islevlerini saglayan Aktif Nesne Kütüphanesi (ANK) aktif nesnelere baglanmaktadir. Adlandirma ve Güvenlik Sunucusu (AGS) Linux çekirdeginin üstünde çalisan bir sunucudur. AGS kalici nesne adini kalici nesne kimligine çevirir, nesneleri yetkisiz erisimlerden korur ve nesnelere erisimde tutarlilik denetimini saglar. Nesne Sunucusu (NS) sistemde nesnelerin uzun ve kisa dönemli bellek arasinda tasinmasindan sorumludur. Sistemi olusturan son bilesen ise nesneler arasinda UNIX IPC yapisi araciligi ile haberlesmeyi saglayan bir haberlesme sistemidir (Nesneler arasi Haberlesme-NH). NH haberlesme ilkelleri senkronize haberlesmeyi mümkün kilmaktadir. Bu islevler bir kütüphane olarak her KGS programina eklenmektedir.

Ilgili Arastirmalar

Bir nesne durumu, davranisi ve kimligi ile tanimlanir (Booch, 1989). Nesne modeli 1970'lerin ortasinda bilgisayar bilimlerinin çesitli alanlarinda ortaya çikmistir. Nesne modeli bazi ilkelerden olusmaktadir; soyutlama, bilgi gizleme, modülerlik, hiyerarsi, kalicilik. Bu ilkelerin bazilari tüm nesneye dayali sistemlerde vardir, bazilari ile bir kisim sistemlerde bulunmaktadir.

Ilk arastirmalar büyük ölçüde kalici programlama ortamlari konusunda olmustu. Bu çalismalarin sonucunda ortaya çikan ürünler arasinda PS-Algol (Cockshot vd., 1984, Atkinson ve dig., 1983), E (Richardson ve Carey, 1989) (Richardson, 1989), Persistent Smalltalk (Eliot vd., 1990) ve Persistent Java (Atkinson vd., 1996) sayilabilir.

Nesne kavramini ilk gerçekleyen nesneye dayali dillerin ardindan nesneye dayali isletim sistemleri gündeme geldi. Nesneye dayali tasarimlari bu isletim sistemlerinin ayirdedici özellikleriydi. Bir isletim sisteminin tasariminda 2 modelden yararlanilabilir: Görev modeli ve nesne modeli (Goscinski, 1991). Iki model arasindaki temel fark islevsel birimlerin organizasyonu ve senkronizasyonundadir.

Görev modeli görevler ve mesajlardan olusur. Tüm sistem etkinlikleri görevler tarafindan gerçeklestirilir. Nesneye dayali modelde ise hizmetler ve kaynaklar nesneler içine gömülmüstür. Nesneye dayali isletim sistemlerindeki ana problem yavasliktir. Diger yandan nesneye dayali isletim sistemleri programlamayi kolaylastirirlar (Almes vd., 1985).

Geleneksel isletim sistemleri genellik ve özellesemeyi dengelemek zorundadir. Genel bir sistem çok degisik programlari çalistirma yetenegine sahipken sadece bazi programlar iyi basari gösterebilir. Varolan sistemlerin yapilari özellestirilmeye uygun olmadigi için sistem karakterindeki ufak degisiklikler bile büyük miktarda programlama yapilmasii gerektirmektedir. Genisletilebilir bir sistem, kullanima yönelik degisikliklerin dinamik olarak yapilabildigi bir sistemdir. Genisletilebilir sistemlere örnek olarak SPIN (Bershad vd., 1995) ve Oberon (Mössenböck ve Wirth 1991) gösterilebilir.

Bir isletim sisteminin dört ana bileseni bellek yönetimi, dosya sistemi, girdi-çikti ve görev yönetimidir. Kalici bir sistemde dosya sistemi ve bellek yönetimi islevleri kalici nesne deposu tarafindan yerine getirilir. (Dasgupta vd., 1991). Bu nedenle kalicilik özelligini destekleyecek bir isletim sisteminin diger isletim sistemlerinden farkli bir tasarimi olmalidir. Kalici nesneleri destekleyen isletim sistemlerine örnek olarak Grasshopper (Dearle vd., 1993)., SOS (Shapiro, 1991 , Shapiro, 1990, Shapiro vd., 1989a, Shapiro vd., 1989b), Mungi (Heiser, 1998) ve Opal (Chase, 1995) gösterilebilir.

KGS ve Programlama

KGS'te programlama bazi farkliliklar disinda UNIX'te program gelistirmeye oldukça benzemektedir. KGS ile saglanan yenilikler sunlardir:

- Kalici nesneler için saglanan destek
- Sunucu programlarinin (aktif nesnelerin) ihtiyaç duyuldugunda kendiliginden yüklenmesi
- Kullanimi kolay olmasina ragmen güçlü ve esnek olan nesneler arasinda haberlesme
- Sistem programciligini kolaylastiran ve hatalari azalmasina yardimci olan bir aktif nesne gelistirme destegi
- Nesnelere erisim yetkileri araciligi ile denetimli erisim

```
/* Kalici nesne tanimlamasi */
$Adlar *pAdAgac;
CapabilityType caV;
/* Yerel Oturumu baslat */
EpsInit(&G1, "demo1", lCapF);
/* Kalici nesneyi yukle */
result=LoadPersistentObject(&G1,(char
**)&pAdAgac, caV, ACMRW);
.
.
/* Yerel oturumu kapat ve kalici
nesnelerdeki degisiklikleri kaydet */
EpsClose(&testSession, COMMIT);
```

Sekil 2. Bir KGS programi

Bir KGS-C programi KGS-C önislemcisi ve C derleyicisi ile derlenir. Program INY ve diger kütüphaneler ile baglanir. Tipik bir KGS-C programi kalici degiskenlerin tanimlanmasi ile baslar (Sekil 2). Ilk birkaç satirda gerekli nesneleri yükleyerek görev ortamini hazirlayan EpsInit() fonksiyonu çagrilir. Arzu edilen islemler tamamlandiktan sonra program, kalici nesneleri kaydeden ve alinmis kaynaklari serbet birakarak kalici görev ortamini kapatan EpsCommit() fonksiyonu çagrilarak sonlandirilir.

KGS-C Programlama Dili

KGS-C, C programlama diline bazi eklemeler yapilarak olusturulmustur. Bu genisletilmis sözdizimini standart C programlama dilinden ayirmak üzere KGS-C olarak adlandirilmistir. Bu gerçeklestirme yöntemi sayesinde programcilar istediklerinde C dilini de kullanabilmektedir. C diline yapilan temel eklenti yeni bir simgenin ("$") sözdizimine eklenmesidir. Bu simge ile programcilar kalici degiskenleri tanimlayabilmektedir (Sekil 2).

Pasif Nesnelerin Tanimlanmasi

Kalici nesnelerin yüklenmesi islemi ya ilgili ilkel (LoadPersistentObject) ile açikca yapilir (Sekil 2), ya da gerekli fonksiyon satirlari programa KGS-C önislemcisi tarafindan eklenir. Kalici nesneleri yüklemek için gereken fonksiyonlarin programci tarafindan çagrilmasi ek bir is gibi görünmekle beraber, yazilimciya nesne yükleme zamanlarini belirleme konusunda sagladigi olanakla degisik amaçli ve karakterli uygulamalari gelistirebilme olanagi yaratmaktadir.

Kalici degisken tipleri herhangi bir basit C tipi veya kullanici tarafindan tanimlanan kayit yapisi tipi olabilir. Her kayit yapisi tanimlamasi kayit yapisinin adini tasiyan bir dosya olarak saklanmalidir (Sekil 3). KGS tasarim ilkelerinin geregi olarak bu tür karmasik yapi tanimlamalarinda özel bir format kullanmak yerine standart C structure/tip kavramlari ve tip denetim mekanizmalari kullanilmistir.

```
Typedef struct{
  char ad[16];
  char *rp;
  char *lp;
}Adlar;
```

Sekil 3. adlar.h

Aktif Nesnelerin Tanimlanmasi

Aktif nesnelerin gelistirilmesi aktif nesne kütüphanesi tarafindan desteklenmektedir. Bu kütüphane programciya istemci taleplerini alip karsilayan bir sunucu döngüsü saglamaktadir (EpsAob). ANK ayni zamanda standart kesinlestirme ve geri dönüs islevlerini de saglar. Aktif nesneler KGS'in bir parçasidir ve bir anlamda kendileri de AGS ve NS karsisinda istemci nesnelerdir .

Programci islevsel bir sunucuya gelistirmek için dört adimdan geçmelidir:

- Statik fonksiyon dizisini doldur (Sekil 4, L1). Bu örnekte, "Ekle" ve "Iste" programci tarafindan gelistirilen fonksiyonlardir.
- Yerel KGS oturumunu "EpsInit" ile baslatan ve sunucu döngüsü için "AobServer" fonksiyonunu çagiran bir ana fonksiyon yaz (Sekil 4, L2).
- Sunucunun kabul edecegi her istek için bir fonksiyon yaz (Sekil 4, L3).
- Programi aktif nesne kütüphanesi (ANK) ve istemci nesne yöneticisini de (INY) kullanarak derle.

```
/*L1:Fonksiyon isaretçi dizisi aktif nesnede
varolan sunucu islemlerini tanimlar */
int   (* f[MAXFUNCTION]) (char *, int) =
{AobCommit, AobRollback, …, Ekle, Iste};
int main(){
/*L2:Yerel oturumu baslat */
EpsInit(&testSession, "warehouse", lCapF);
/* Aktif nesne istemci isteklerini dinler ve
talepleri karsilar */
AobServer(f);
/* Yerel oturumu kapat ve kalici
nesnelerdeki degisiklikleri kesinlestir */
EpsClose(&testSession, COMMIT);
}
/*L3:Bir sunucu fonksiyonun gerçeklenmesi */
int Ekle(){…
}
```

Sekil 4. KGS aktif nesnesi kodu

Artik aktif nesne hizmet vermeye hazirdir. Kendisine yönelik bir istek geldiginde NS tarafindan çalisir duruma getirilerek hizmet vermesi saglanacaktir.

KGS-C Önislemcisi

KGS-C C derleyicisi ile beraber kullanilan bir önislemci ve istemci nesne yöneticisi araciligi ile gerçeklenmistir. KGS-C programlari "epsc" uzantisi tasirlar. Önislemci "epsc" uzantili kaynak kodunu standart C derleyici ile islenebilen "c" uzantili bir dosyaya dönüstürür.

Ilk isleme döngüsünde, önislemci program kodunu ayristirarak kalici degisken tanimlarini normal C tanimlari ile degistirir. Bu döngü sonunda kalici degiskenlere ait tip, ve boyut gibi bilgilerin saklandigi bir degiskenler tablosu olusmustur (Sekil 5). Kayit yapisi gibi karmasik tiplerin boyutlari ve yapisi bu kayit yapisini tanimlayan include dosyasinin yorumlanmasi sonucunda belirlenmektedir. KGS-C önislemcisi ve çalisma ortami bu include dosyayi kullanarak kalici nesnenin özelliklerini belirler.

Ad	Tip	Isaretçi	Adet	Boyut
pAgac	Adlar	E	1	24
caV	Capability	H	1	50
.	.	.	.	.

Sekil 5. Degiskenler tablosu

Ikinci isleme döngüsünde ise, eger önislemci seçenekleri uygunsa ilk asamada hazirlanan degiskenler tablosu kullanilarak kalici degiskenlerin yüklenmesini saglayacak ilkeller kaynak koduna eklenir.

KGS Tasarimi

Bu bölümde sistemin mimarisi açiklanmaktadir. KGS varolan bir isletim sistemi üzerine insa kurulmustur. Elde edilebilirlig, açik kaynak kod politikasi ve gelistirme araçlarini rahatlikla saglanabilmesi nedeniyle gelistirme platformu olarak Linux isletim sistemi seçilmistir.

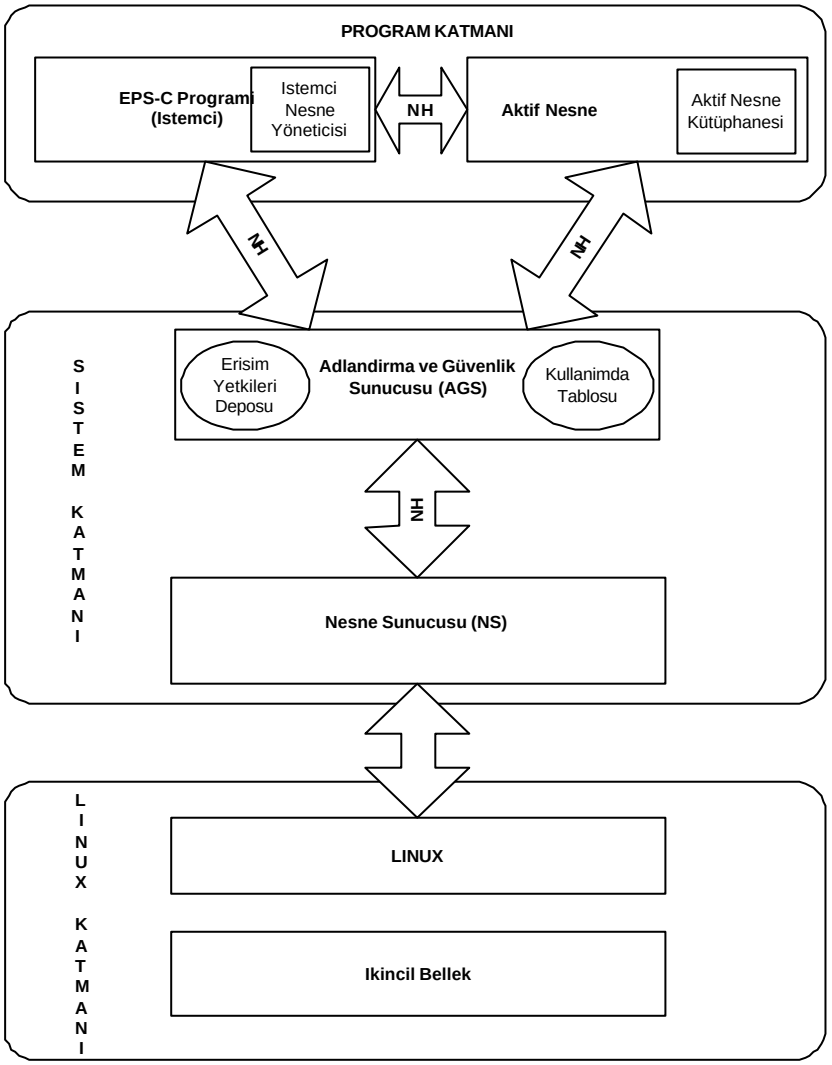
KGS Sistem Mimarisi

KGS gereksinimler dogrultusunda genisletilmek amaciyla tasarlanmis olan çok katmanli bir mimariye sahiptir (Sekil 6). Temel sistem üç bilesenden olusmaktadir; Adlandirma ve Koruma sunucusu (AGS), Nesne Sunucusu (NS) ve Istemci Nesne Yöneticisi (INY). Ayrica bu üç bileseni birbirine baglayan bir haberlesme olan Nesneler arasi Haberlesme (NH) hizmeti vardir. Bu bilesenler sistemin temelini

olustururlar. Sistemden daha farkli hizmetler almak için ise aktif nesnelerden yararlanilmaktadir.

Nesneler arasindaki iletisim yüksek düzeyli haberlesme ilkelleri ile saglanmaktadir. Mesajlajma hizmeti Linux IPC ayrintilarini programcidan saklayarak nesnelerarasi haberlesmeyi saglar.

Nesneler arasi haberlesme hizmeti Unix IPC mekanizmasinin üzerine kurulmustur. Saglanan üst düzey ilkeller hem sistem hem de uygulama programi gelistirilmesinde kullanilabilmektedir (Ek 1).



Sekil 6. KGS tasarimi

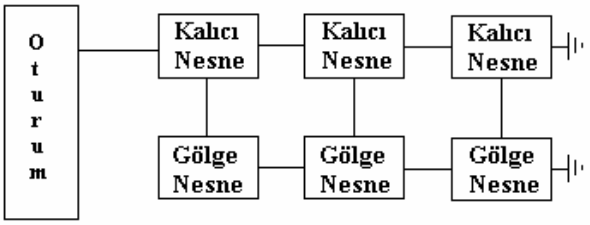
Istemci Nesne Yöneticisi

Istemci Nesne Yöneticisi (INY) her KGS programi ve aktif nesne tarafindan kullanilir. INY, sistemin karmasikligini kullanicidan saklamaya yardimci olur. Bazi sistem ilkelleri de kismen veya tamamen INY içinde gerçeklenmistir. INY ayni zamanda tutarlilik denetimi ve adres dönüsümü gibi hizmetlerin yerel bölümlerini de gerçeklestirir. Bu hizmetlerin diger bölümleri ise KGS sunucularinda ve aktif nesnelerde gerçeklenmistir.

INY KGS'in yerel yönetim kavramini ve kalici nesnelerin yerel saklama alanini olusturan istemci oturumunu yaratir ve yönetir (Sekil 7).

Bir istemci programi çalismaya baslayinca görevin adres uzayi içinde yeni bir oturum olusturulur. Görev için gereken kalici nesneler programcinin müdahalesi olmadan yüklenir ve islenirler.
Oturum'u tanimlayan ana bilgi yapisi olan Oturum nesnesi aktif nesne, yüklü nesneler gibi görev ile ilgili bilgileri saklar. Oturum nesnesi iki adet paralel bagli listeden olusur (Sekil 7). Iki listedeki paralel dügümler ayni nesnenin ilk ve degismis hallerini saklayarak gerektiginde kesinlestirme ve geri dönüs islemlerine olanak saglarlar.

INY ayni zamanda görev tarafindan daha önce edinilen erisim yetkisi'leri saklayan yerel erisim yetkileri deposunu da yönetir. Bir nesne için istekte bulunulacagi zaman önce yerel depoda ilgili nesne için erisim yetkisi olup olmadigi kontrol edilir. Eger yerel depoda yoksa AGS'ten talepte bulunulabilir.



Sekil 7. Oturum mimarisi

Adres Dönüsümü

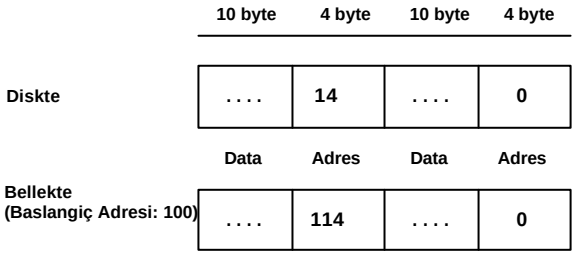
Birincil bellege NS tarafindan yüklenen ve nesneler arasi haberlesme sistemi araciligi ile göreve iletilen kalici nesnelerin kullanilabilmesi için bir isleme daha ihtiyaci vardir; kalici isaretçi adreslerinin yerel adreslere dönüstürülmesi. Kalici degiskenlerin veri bölümleri bellege yüklenirken özel bir isleme gerek yoktur, ancak isatetçi içeriklerinin yerel adreslere dönüstürülmesi gerekmektedir.

Adres dönüsümü, nesne tipine bagli olarak INY tarafindan nesnelerin yüklenmesi sirasinda gerçeklestirilir. Bazi tasarimlarda, tembel dönüsüm de denilen bir yöntemle adres dönüsümü nesnenin yüklendigi anda degil de nesneye ilk erisim gerçeklestiginde yapilarak yükleme süresinin kisaltilmasi amaçlanmistir. Ancak, KGS'te gerekli islemlerin tümünün yükleme aninda gerçeklestirilir, çünkü KGS programcilara nesne yükleme zamanlarini belirleme konusunda bir esneklik saglamistir. Programci yükleme zamanina, performans ve yazilimin özelligini dikkate alarak kendisi karar verebilir; böylelikle degisik ihtiyaçlari karsilamasi mümkün olacaktir. Isaretçi degerleri bir kez degistirildikten sonra kalici veri görev içinden kullanilmaya hazirdir, ve sadece kalici ortamin bulundugu ikincil bellege yeniden yazilacaklari zaman kalici adrese dönüstürülmeleri gerekecektir.

Adres dönüsümü iki ana asamadan olusmaktadir; Ilk adim, nesnenin yapisinin ayristirarak anlamaktir. Bu adim karmasik nesneler için include (.h) dosyalari incelenerek gerçeklestirilir. Asagida görülen örnek veri yapisinin adres dönüsümü sonuçlarini Sekil 8'de görülmektedir.

```
typedef struct {
        char arr1[10];
        int *p1;
}TestObjectType;
```

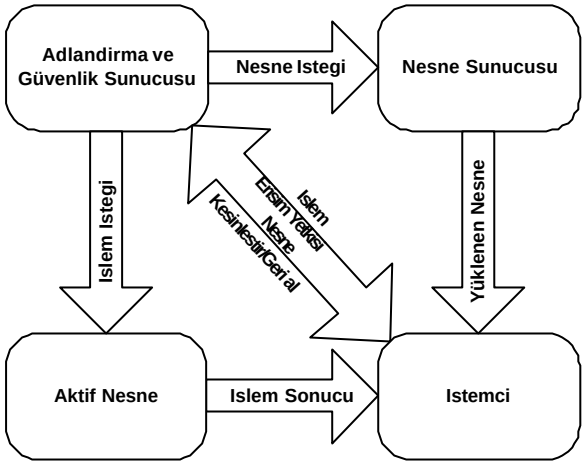
Ikinci asamada, daha önce elde edilen yapi bilgileri kullanilarak kalici adres degerleri görevin bellegindeki adres degerlerine dönüstürülür. Adres dönüsümüne olanak saglayan özel bellek uzaylarinin yönetimi PMalloc ve PFree ilkelleri tarafindan saglanir.



Sekil. 8 Adres dönüsümü

Adlandirma ve Güvenlik Sunucusu

Adlandirma ve Güvenlik Sunucusu (AGS) geri planda çalisan bir sunucu olarak gerçeklenmistir. AGS nesne erisimlerinin güvenlik ve senkronizasyonundan sorumludur. Nesne erisimi için yapilan her istek AGS tarafindan karsilanir (Sekil 9).



Sekil 9. KGS mesaj akisi

Kullanimda tablosu yüklü/kullanilan nesnelerin kayitlarini tutar. Tablonun tek kopyasi vardir, böylece erisimler için gereken denetim baska kaynaklara basvurmaya gerek kalmadan bu tablo üzerindeki bilgiler ile yapilabilmektedir. Kullanimda tablosu ayni zamanda tüm erisim yetkisi bilgilerini de depolar. Tablonun boyutlari ihtiyaca göre dinamik olarak degisebilmektedir.

AGS'e isteklerin gönderilmesi ve yanitlarin alinmasi için nesneler arasi haberlesme sistemi (NH) kullanilir. Aktif nesne hizmet istekleri de AGS tarafindan ele alinir, ancak talep edilen islemin özellikleri aktif nesne tarafindan bilindigi için sadece erisim yetkileri denetimi yapilir ve islem gerçeklestirilmek üzere aktif nesneye yönlendirilir.

Nesne Adlarinin Çözümlenmesi

Kalici nesne isimleri iki bölümden olusurlar: yaratan nesne adi ve kalici nesne adi. Yaratici nesne adi, kalici nesnenin ilk kez tanimlandigi program adidir. Kalici nesne adi ise, kalici nesnenin program içindeki degisken olarak varoldugu addir. Uzun ömürleri ve genis kapsama alanlari nedeniyle kalici nesne adlarinin tekrar kullanilma olasiligi artmaktadir. Bu iki seviyeli isimlendirme kalici nesnelerin birbirinden ayrilmasini kolaylastirmakta, ayni zamanda kalici nesne adlarinin çakisma olasiligini da azaltmaktadir.

Bir nesne için adiyla talepte bulunulamaz, öncelikle yerel erisim yetkileri deposundan veya AGS'ten bu nesnenin erisim yetkileri bilgisi alinmalidir. AGS bu nesne için gereken bilgileri talep eden nesneye geri gönderir. Bir kez elde edildikten sonra yerel erisim yetkileri deposunda saklanan nesne bilgileri adla yapilan nesne erisim isteklerinde çözümleme için AGS'ye basvurmayi gereksiz kilar. Yerel depodaki nesneler için ayni isimden uzun bir tamsayi olan kimlige dönüstürme islemi yerel olarak gerçeklestirilir. Bu yapiyi bir örnek ile biraz daha açiklayalim: depo.epsc adinda bir program içinde asagidaki kalici degisken tanimli olsun

```
$int stokSay
```

Kalici nesne adi "depo.stokSay", kalici kimligi de 100100 gibi KGS-C tarafindan atanmis herhangi bir uzun tamsayi olacaktir.

AGS bilgi istenilen bir nesneyi kendi erisim yetkisi deposunda arastirir. Arama önce nesne adi ve yaratici adi ile yapilir. Eger aranilan nesneyi bulamazsa, ikinci kez sadece nesne adi ile arama yapar. Bu yaklasimla bir yandan nesne isimlerinin çakismasi önlenmeye çalisilirken, diger yandan da sadece nesne adinin bilinmesi gibi özel durumlarin da çözümü aranmistir.

Nesne adi çözümleme islemi bir nesne talep edildiginde, ancak erisim yetkisi yerel görevin erisim yetkisi deposunda bulunmadigi zaman gerçeklestirilir. Bu durumda erisim yetkisi AGS'ten talep edilir ki, o da kendi erisim yetkisi deposundan biraz önce anlattigimiz algoritma ile arar, ve bulursa talep eden göreve iletir. Erisim yetkisi'nin yerel depoda bulunma islemi de ayni algoritma ile gerçeklestirilir.

Erisim Denetimi

Bir istek uygun bulundugunda Kullanimda tablosuna yüklenen nesneye ait bilgileri tasiyan bir kayit yazilir. Eger istek bir nesne yüklenmesi için ise NS'e iletilir. Bu durumda yükleme isleminin sonucu NS tarafindan istemciye gönderilir. Bir istemci ayni zamanda birden fazla kalici nesneye erisebilir. KGS'deki erisim denetimi ve güvenlik mekanizmalari tüm

nesneyi kapsamaktadir. Bir nesnenin alt bölümlerini ayri ayri denetleyebilecek daha küçük kapsamli bir güvenlik yoktur.

Erisim tutarliligi konusunda pasif (sadece veri) ve aktif (veri ve metodlar) için degisiklikler vardir. Aktif nesne ayni anda tek istege yanit verebilecegi için senkronizasyon problemi olusmaz. Pasif bir nesne ise birden fazla istemci, yani KGS-C programi tarafindan paylasilabilir, bu nedenle pasif nesnelere erisimde erisim denetimi konusu özel olarak ele alinmak zorundadir. Pasif nesne için yapilan bir istek geçerlilik denetiminden geçtiginde, AGS Kullanimda tablosundaki nesne için eklenen satiri Tablo 1'deki degerlerle günceller. Ayni nesne için yeni bir istek geldiginde halen kayitli olan erisim düzeyi ve durum degisim kurallari kullanilarak istegin karsilanip karsilanamayacagi belirlenir. Eger erisim mümkün degilse AGS istemci nesneye bir red mesaji gönderir. Bir nesneye erisim hakki kazanmis olan görev isi bittiginde AGS'e bildirerek o nesne için erisim düzeyini sifirlar, yani Kullanimda tablosundaki Durum alanini "no" olarak degistirir.

Tablo 1. Erisim haklari geçis tablosu

no→ro	ro→no	sa→no	rw→no	rwd→no
no→sa	ro→sa	sa→ro	rw→ro	rwd→ro
no→rw	ro→rw	sa→sa		
no→rwd	ro→rwd			

(NO access, ReadOnly,Shared Access, ReadWrite,ReadWriteDelete)

Nesne Sunucusu (NS)

Nesne sunucusu(NS), KGS'in kalici nesne deposudur.Fiziksel olarak ayri bir sunucu görev olarak gerçeklenmistir. Nesne sunucusu aktif ve pasif nesneleri yükleme ve saklama islemlerini gerçeklestirir.

Bir yükleme islemi istegi aldiginda, NS önce istegin bir aktif nesne için mi yoksa bir pasif nesne için mi oldugunu belirler, sonra da uygun yükleme islemini gerçeklestirir. NS sadece AGS'ten gelen istekleri karsiladigi için erisim yetkileri ve senkronizasyon konularinda bir çözüme ihtiyaç duymaz.

Bellekteki birden fazla nesnenin tutarliligi gölge nesne mekanizmasi ile gerçeklestirilir. Tüm nesneler bellekte orjinal ve degismis halleri olmak üzere iki sekilde saklanir. Bir kesinlestirme islemi gerçeklestiginde degismis versiyonlar orjinal degerlerin üzerine kopyalanir ve tüm nesne degerleri dosya sistemine kaydedilir. Bu mekanizma araciligi ile ayni istemcinin yerel adres uzayinda bulunan birden fazla pasif nesnenin tutarliligi korunabilir.

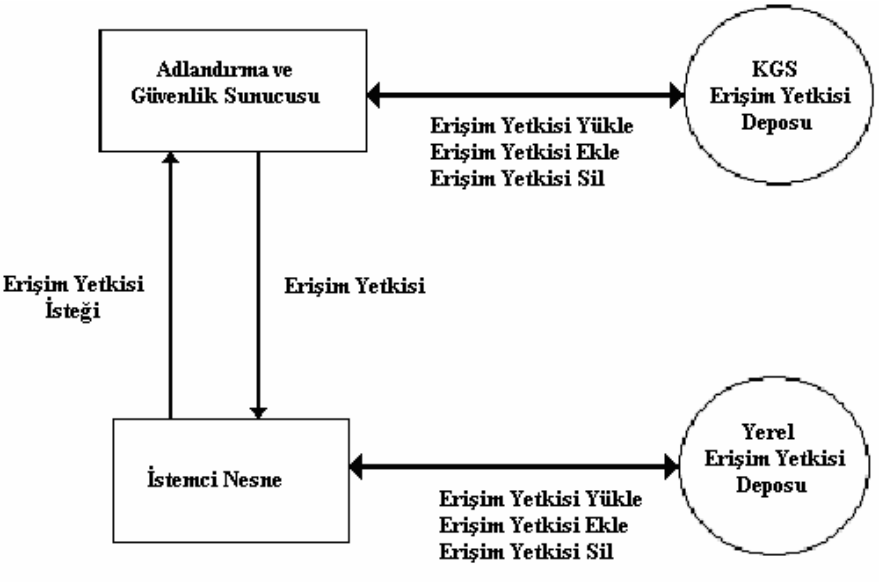
Aktif nesnelere yapilan taleplerde ise farkli konular öne çikar. Bir istemci nesne aktif nesneden yaptigi istek kapsaminda bir deger degisikligine neden olursa bu degisiklikleri kesinlestirmek için de kesinlestir/geriye döndür mesajlari gönderebilir.

Tutarsiz bir veritabani yaratma riskini azaltmak için gölge dosya teknigi kullanilmistir. Gölge dosya tekniginde degisen bir nesneye ait bilgileri içeren bir dosya önce farkli bir dosyaya yazilir. Dosyaya yazma islemi tamamlandiktan sonra tüm dosya orjinal dosyaya kopyalanir. Bu yöntem yazma isleminin basarimini düsürdügü için bir seçenek olarak saglanmistir ve NS çalistirilirken bir parametre ile verilir.

Kalici nesneler uzun dönemli olarak disk üzerinde saklanirlar, ve NS tarafindan birincil bellege yüklenirler. Birincil bellege yüklenen bir pasif nesne kök nesnesi ile temsil edilir. Kök nesne, program içinde bir degiskenle iliskilendirilmistir ve program komutlari tarafindan üzerinde islem yapilabilir (Sekil 2, pAdAgac). Diger nesneler kök nesne ile isaretciler araciligi ile baglantilidir. Istemci nesneler talep ettikleri nesnelerin kök nesnesinin fiziksel adresine sahip olurlar, ve bu adres araciligi ile pasif nesne içindeki diger bilgilere ulasirlar.

Her kalici nesne grubu ayri bir dosyada saklanir ve dosya içinde homojen bir yapi vardir. Bu gruplar kalici veritabanlari olarak adlandirilirlar. Kalici nesnenin tip bilgisi kalici nesne dosyasinin baslik bilgileri bölümünde saklanir.Tamsayi ve karakter gibi basit veri

tipleri dogrudan baslik bilgileri içine konulurken, karmasik kullanici tip tanimlari ayri dosyalarda saklanip baslik bölümünde atifta bulunulur.



Sekil 10. Erisim yetkileri depolari

Aktif Nesne Gerçeklenmesi

Aktif nesneler (AOB) KGS'nin bir parçasi degildir. Aktif nesneler, kalici sunucu görevleridir ve kendileri de ayni zamanda programcilar tarafindan yaratilan birer istemci nesnedir.

Aktif nesnelerin önemli bir özelligi, KGS tarafindan desteklenmeleridir. KGS önceden gelistirilmis olan bir kalici aktif nesneyi ihtiyaç oldugunda seffaf bir sekilde yükler. Aktif nesnelere erisimin yetki denetimi pasif nesnelerde oldugu gibi, AGS tarafindan gerçeklestirilir. Aktif nesneler istemci nesne yöneticisini (INY) ve nesnelerarasi haberlesme sistemini (NH) kullanirlar. Istemci nesnelerden gelen istek mesajlarini karsilayan bir arayüz islevi aktif nesne kütüphanesi tarafindan saglanmaktadir. Bu kütüphane haberlesme ve erisim tutarliligi hizmetlerini ve sunucu islem döngüsünü içerir.

KGS Güvenlik Mimarisi

KGS'de nesnelere erisim güvenligi erisim yetkilerine dayanmaktadir.

KGS erisim yetkisi yapisi ID'ler, kalici nesne adi, bu erisim yetkisi ile verilen haklar ve CRC gibi alanlardan olusur. Her yapilan istekle beraber istemci tarafindan AGS'ye sunulan her erisim yetkisinin önce istemci nesne için olup olmadigi kontrol edilir.

Her istemci nesnenin AGS tarafindan isletilen erisim yetkisi deposunun yaninda kendi deposu vardir (sekil 10).
AGS her erisim yetkisi'nin geçerliligini ve talep edilen nesnenin erisim kisitlarini kontrol ettikten sonra sagladigi yetkilerin geregi islemleri gerçeklestirir. Yetki denetimi üç asamali olarak gerçeklestirilir; Iletilen erisim yetkisi istemci nesnenin kullanimina uygun mu, iletilen erisim yetkisi talep edilen nesne için mi ve erisim yetkisi içerigi orjinal mi? Böylece istemcilerin sadece kendi kullanimlarina açik olan erisim yetkisileri kullanmalari garanti altina alinmistir.

Basarim

KGS'nin basarimini iki degisik açidan incelemek yararli olacaktir; program gelistirme ve program çalistirma.

Ölçüm için gelistirilen programlar bir agaç yapisini yükler, istenildiginde alfabetik olarak listeler ve yeni dügümler ekler. Ölçüm için agaç yapisinin ortalama yükleme zamani program

içine eklenen küçük bir kod parçasi ile mikrosaniye cinsinden hesaplanmistir.

Demo1.c programi KGS üzerinde çalistirilmistir. Demo11.c programi ise tüm fonksiyonlari içinde barindirir ve Linux üzerinde çalismaktadir. Tablo 2'de program satir sayisini Tablo 3 ve 4'de ise yükleme süresi ile veri gösterme süresi ölçümlerini bulacaksiniz.

Tablo 2'deki kazanç demo1.c'de nesne yükleme ve kaydetme ilkellerinin KGS tarafindan saglanmasi, buna karsin demo11.c'de bu fonksiyonlarin programci tarafindan gerçeklenmesinden kaynaklanmaktadir. Iki programin kaynak kodlari, tablo 2'de gösterilen farki yaratan ilkeller ve fonksiyonlar farkli renkte gösterilmis olarak, Ek-2'de mevcuttur.

Tablo 2. KGS programi satir sayisi karsilastirmali ölçümü

Program Adi	Program Satir Sayisi
Demo1.c	130
Demo11.c	171
KAZANÇ	**%24**

Tablo 3'de görülen yükleme zamanlari arasindaki fark demo1.c'nin istemci/sunucu mimarisinde çalismasina ve ek güvenlik katmanina karsin, demo11.c'nin isletim sistemi üzerinde çalisan ve tüm fonksiyonlari kendisi gerçekleyen bir program olmasindan kaynaklanmaktadir. Yükleme sonrasinda nesne üzerinde yapilacak islemler konusunda ise KGS ile yazilan program ve digeri arasinda hiç bir yöntem farki yoktur. Yani, KGS üzerinde yazilan bir programda nesneler bir kez yüklendiginde diger program degiskenleri ile ayni sekilde islem görmektedirler (Tablo 3). Bu özellik nedeniyle örnek programlarimiz demo1.c ve demo11.c'nin nesne yükleme ve yazma disindaki bölümleri aynidir.

Tablo 3. KGS programi veri gösterim süresi karsilastirmali ölçümü

Program Adi	Pentium (ms)	Pentium IV (ms)
Demo1.c	4584	95
Demo11.c	4603	109

Tablo 4'de görülen ölçümlerde dikkat çekici olan nokta ölçümün yapildigi bilgisayar sisteminin performansina bagli olarak sürelerin önemli ölçüde kisalmasidir. Bilgisayar sistemlerinin daha da hizlanmasi ile KGS'nin kullanimi ile olusan yükleme performansi dezavantaji önemini yitirmekte, buna karsin programlamayi kolaylastirma özelligi devam etmektedir.

Tablo 4. KGS programi nesne yükleme süresi karsilastirmali ölçümü

Program Adi	Pentium (ms)	Pentium II (ms)	Pentium IV (ms)
demo1.c	3508	797	199
demo11.c	1907	352	208

Sonuç

KGS genisletilebilir sistemler için kalici nesnelere dayanan yeni bir model önermektedir. Bu modelde veri isleme yetenegine sahip olan görevler ve veriler kalici nesneler olarak ele alinmakta ve tek bir arayüz ile islenmektedir. Sistem tanimi aktif, pasif ve statik nesne tanimlari ile rekürsif olarak gelistirilebilmektedir. Bu basit ama güçlü model sistem tasariminin anlasilmasini ve yeni yazilimlarin gelistirilmesini kolaylastirmaktadir. KGS'de sunucu islevlerinin gelistirilmesi standart bir arayüz ve kütüphanelerle destekleniyor, böylece KGS, programcilarin birer sunucu olan aktif nesneleri geleneksel sunucu programlamasindan farkli olarak daha kisa sürede gelistirebilmelerini sagliyor.

Gelisen donanim yetenekleri sayesinde KGS gibi sistemlerle saglanan yeni olanaklarin sistem basariminda neden olduklari olumsuz etki büyük ölçüde ortadan kalkmaktadir. Bu nedenle sistem karmasikligini azaltici ve yazilim gelistirmeyi hizlandiracak yeni çalismalarin giderek yayginlasacagi beklenilmektedir. Program gelistirmeyi kolaylastirma konusunda önemli bir araç olan gelistirme araçlari da bundan sonra yapilabilecek bir çalismadir.

Kaynaklar

Atkinson, M. P., Bailey, P. J., Chisholm, K. J., Cockshott, P. W. ve Morrison, R. (1983). An Approach to Persistent Programming, *The Computer Journal*, **26** , 360 - 365.

Atkinson, M.P., Daynès, L., Jordan, M.J., Printezis, T. ve Spence, S. (1996). An Orthogonally Persistent Java, *ACM SIGMOD Record*, **25**, 4, 68 – 75.

Bershad, B. N., Savage, S., Pardyak, P., Sirer, E. G., Fiuczynski, M. E., Becker, D., Chambers, C., Eggers, S. (1995). Extensibility, Safety and Performance in the SPIN Operating System, *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, 267-284, ABD

Booch, G. (1989). *Object Oriented Design*, 580 sh., The Benjamin/Cummins Publishing Company Inc., California.

Chase, J. S. (1995). An Operating System Structure for Wide-Address Architectures, *Doktora Tezi*, University of Washington, Seattle, WA.

Cockshot, W. P., Atkinson, M. P. ve Chisholm, K. J. (1984). Persistent Object Management System, *Software- Practice and Experience*, **14**, 49-71.

Dasgupta, P., LeBlanc, R. J., Ahamad, M. ve Ramachandran, U. (1991). The Clouds Distributed Operating System, *IEEE Computer*, **18**, 34-43.

Dearle, A., de Bona, R., Farrow, J., Henskens, F., Lindstrom, A. ve Rosenberg, J. (1993). Grasshopper: An orthogonally persistent operating system, *Teknik Rapor/GH-10*, University of Adelaide, Adelaide, Australia.

Dearle, A. ve Hulse, D. (2000). Operating system support for persistent systems: past, present and future, Software - Practice and Experience, *Special Issue on Persistent Object Systems*, **30**, 4, 295-324.

Eliot, J. ve Moss, B. (1990). Design of the Mneme Persistent Object Store, *ACM Transactions in Information Systems*, **8**, 2, 103-109.

Heiser, G., Elphinstone, K., Vochteloo, J., Russell, S. ve Liedtke, J. (1998). Implementation and Performance of the Mungi Single-Address-Space Operating System, *Software: Practice & Experience*, **28**, 901-928.

Gibbs, W. W. (1994). Software's Chronic Crisis', *Scientific American*, Eylül, 72-81.

Mössenböck, H., ve Wirth, N. (1991). The Programming Language Oberon-2, *Structured Programming*, **12**, 179-195.

Kemikli, E. ve Erdogan, N. (1997). Persistent Operating Systems., *Proceedings of the 12th International Symposium on Computer and Information Sciences (ISCIS XII)*, 76 – 84, Antalya.

Kemikli, E. ve Erdogan, N. (1998). Persistent Operating Systems, *Proceedings of the 9th Mediterranean Electrotechnical Conference (MELECON 98)*, 1304 – 1307, Tel-Aviv.

Kemikli, E. ve Erdogan, N. (1999). Extendible Persistent System, *Proceedings of 3rd WSES/IEEE/IMCS Inernational Conference on Circuits, Systems, Communications and Computers (CSCC'99)*, 4971 – 4974, Atina.

Kemikli, E. ve Erdogan, N. (2002). Augmenting Object Persistency Paradigm for Faster Server Development, *Lecture Notes in Computer Science*, **2457**, 327-335

Richardson, J. E. ve Carey, M. J. (1989). Persistence in the E Language: Issues and Implementation. *Software-Practice and Experience*, **19**, 1115-1150.

Richardson, J. E. (1989). E: A Persistent Systems Implementation Language, *Doktora Tezi*, University of Wisconsin-Madison, Wisconsin.

Shapiro, M. (1990). Object-Support Operating Systems, *Position paper for Workshop on Operating Systems and Object Orientation at ECOOP-OOPSLA*, Ottawa.

Shapiro, M. (1991). Soul: an object-oriented OS framework for object support, *Proceedings of Operating Systems for the nineties and Beyond workshop*, 251 – 255, Dagstuhl Castle.

Shapiro, M., Gourhant, Y., Habert, S., Mosseri, L., Ruffin, M. and Valot, C. (1989). SOS: An Object Oriented Operating System- Assessment and Perspectives, *Computing Systems*, **2**, 287-337.

Shapiro, M., Gautron, F. ve Mosseri, L. (1989). Persistence and Migration for C++ Objects. *Proceedings of the European Conference on Object-Oriented Programming (ECOOP`89)*, 191 – 204, Nottingham.

Tanenbaum, A. S. (1992). *Modern Operating Systems*, 728 sh., Prentice Hall Internatio nal, New Jersey.