Gerçek Zamanlı Sistemlerde UML Uygulamaları

Ersel ERCEK[1] Sevda ERDOĞDU[2] Mehmet Fatih ULUAT[3]

[1,2,3]ASELSAN A.Ş. Mehmet Akif Ersoy Mahallesi 16. Cad. No:16,
Yenimahalle – ANKARA

[1]ercek@mst.aselsan.com.tr [2]erdogdu@mst.aselsan.com.tr
[3]mfuluat@mst.aselsan.com.tr

Özet

Gerçek zamanlı uygulamalar ve gömülü sistemler fonksiyonellik, fiyat, boyut, performans, yaygınlık vb. açılarından değerlendirildiğinde; günlük hayattan, tıp, telekomünikasyon, havacılık, uzay ve savunma sanayine kadar tüm alanlarda 8 bit işlemci tabanlı sistemlerden, yüksek performanslı işlemci ağlarından oluşan sistemlere kadar çeşitlilik gösterirler. Her gün ortalama olarak 70 adediyle karşı karşıya geldiğimiz gerçek zamanlı ve gömülü sistemler, sistem mühendisliği koordinasyonunda yazılım, donanım, mekanik, test gibi birden fazla uzmanlık disiplininin ortaklaşa gerçekleştirdiği ürünlerdir. Bu bildiride Kaideye Monteli Stinger (KMS) Sistemi Atış Kontrol Bilgisayarı (AKB) içindeki PowerPC işlemci kartları üzerinde koşan uygulama yazılımlarının; sistem analiz, tasarım ve geliştirme aşamalarında ASELSAN A.Ş. yazılım geliştirme yönergeleri çerçevesinde uygulanan UML (Unified Modeling Language) tabanlı modelleme süreci sunulacaktır.

Abstract

Real-time applications and embedded systems vary in functionality, cost, size, performance, ubiquity, etc. in all areas from daily life, medicine, telecommunication, aeronautics, space and defence industry; from 8 bit processor based systems to high performance processor network based systems. We are in touch with approximately 70 of real-time applications and embedded system applications everyday. These systems are products of a multi-disciplined approach i.e. software, hardware, mechanical, test efforts coordinated by systems engineering. In this paper, UML based modelling process applied during the system analysis, design and development phases of the application softwares run on the PowerPC processor of Pedestal Mounted Air Defence System's Fire Control Computer is presented.

1. Giriş

Günümüzde 15 milyondan fazla mikro yongayı ve günlük hayatımızda her gün ortalama olarak 70 adet mikro işlemci kontrolündeki cihazı ihtiyaçlarımızı karşılamak için kullanmaktayız. Tıp, telekomünikasyon, havacılık, uzay, savunma sanayi vb. birçok alanda hayatımıza giren gerçek zamanlı ve gömülü sistem uygulamaları, beklenen performans gereklerini ve zaman kritik görevleri herhangi bir hata oluşmasına izin vermeden yerine getirmektedir.

Gerçek zamanlı sistemler, senkron / asenkron olarak çalışan ve performans sınırlamaları olan birimlerin bir araya gelmesinden oluşur ve beklenen görev kritik performans gereklerini güvenilir, doğru, sağlam ve uygun zamanlamaya sahip çıktıları ile yerine getirirler. Gerçek zamanlı sistemler zamanlama, yerindelik açısından değerlendirildiğinde gevşek gerçek zamanlı (Soft Real Time System) ve katı gerçek zamanlı (Hard Real Time System) olarak sınıflandırılabilirler. Gevşek gerçek zamanlı sistemlerde oluşan hatalar nedeni ile zamanlama kısıtlamaları uygulamanın belirli bir periyodunda sağlanamazsa performans düşer fakat sistem kullanım dışı kalmaz. Kullanılan veriler halen geçerli olabilir. Fakat katı gerçek zamanlı sistemlerde herhangi bir hata nedeni ile geciken veri kullanışsız bir veridir ve tüm sistemin kullanım dışı kalmasına sebep olarak felakete yol açabilir.

Bu açıdan değerlendirildiğinde, alçak irtifa hava savunma sistemi olarak kritik görevleri yerine getiren katı gerçek zamanlı bir sistem olan Kaideye Monteli Stinger (KMS) Sistemi'nin en önemli parçası, sistem içersindeki elektro-optik, servo ve silah gibi alt sistemlerin, sensörlerin, eğleyicilerin (actuator) kontrolünü sağlayan Atış Kontrol Bilgisayarı'dır (AKB). AKB, VME (Versa Module European) Bus tabanlı ve askeri ortam koşullarına dayanıklı bir kasa içerisine yerleştirilmiş 2 adet işlemci kartı, KMS alt sistem birimleri ile arayüzleri sağlayan I/O kartlarının birlikte çalışmasından oluşur.

KMS Sistemleri; **Zıpkın ve Atılgan (Şekil 1)** ana silah olarak **Stinger füzesini** kullanan ilk milli alçak irtifa hava savunma sistemidir. Tüm işlevlerini **bilgisayar denetimli** olarak gerçekleştiren KMS Sistemi, hava hedeflerini gece/gündüz **her türlü hava koşulunda otomatik** olarak yönlenerek takip etmekte ve nişancının komut vermesinin ardından hedefleri imha edebilmektedir.

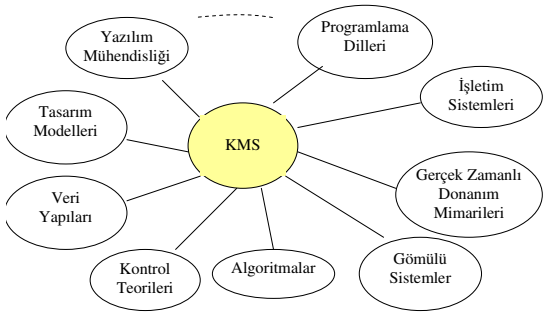
Zıpkın

Atılgan

Şekil 1. KMS Sistemi Konfigürasyonları

KMS Sistemi, sistem mühendisliği koordinasyonunda birden fazla uzmanlık disiplininin (Şekil 2) ortaklaşa çalışmasıyla gerçekleştirilmiş bir üründür. KMS Sistemi yaşam döngüsünde, bir arada çalışan uzmanlık disiplinlerinin koordinasyonunu sağlamak için aşağıdaki süreçler paralel olarak yürütülmüştür.

- Kalite Güvencesi Süreci (KGS)
- Proje Yönetim Süreci (PYS)
- Sistem Geliştirme Süreci (SGS)
- Yazılım Yönetim Süreci (YYS)
- Yazılım Geliştirme Süreci (YGS)
- Yazılım Alt Yüklenici Yönetim Süreci (YAYS)
- Yazılım Konfigürasyon Yönetimi Süreci (YKYS)



Şekil 2. KMS Sistemi Uzmanlık Disiplinlerine Örnekler

2. Gerçek Zamanlı Gömülü Sistemler ve Modelleme

Gömülü bilgisayar sistemleri, genellikle masa üstü bilgisayarlarda olduğu gibi geniş bellek alanları, yüksek işlemci hızları ve sabit disklere sahip değildir. Amaç, her bir gömülü sistem için uygun maliyette maksimum performans ve fonksiyonellik gereğini sağlamak, tekrarlanır maliyeti olabildiğince aşağı çekmektir. Gömülü sistemlerdeki yazılımlar sistemin bir parçası olarak sunulurken, tekrarlanır maliyet olarak değerlendirilmezler. Tüm maliyet; analiz, tasarım ve bakım-idame aşamalarında gruplanır. Donanım mimarilerinin belirlenmesi aşamasında yapılan kısıntılar, kapsamlı performans optimizasyonları ve karmaşık algoritma ihtiyaçları ile doğrudan yazılım maliyetlerini artırabilir ve sistemin genişleyebilirlik, yerindelik, sağlamlık, fonksiyonellik ve performans özelliklerini sınırlayabilirler. KMS Sistemi, katı gerçek zamanlı uygulama isterleri de değerlendirildiğine, analiz aşamasında alınan doğru yazılım ve donanım mimarisi ön tasarım kararları ile yüksek performanslı, maliyet etkin, genişleyebilir, sağlam ve tekrar kullanılabilir bir altyapı üzerine kurulmuştur.

Benzer nedenlerle, gerçek zamanlı ve gömülü sistemler sadece yazılım yönleri ile değil, sistem ve müşteri isterleri açısından da değerlendirilmelidir. Bu da farklı uzmanlık disiplinlerinin bir arada çalışması ve ortak bir terminoloji ihtiyacına neden olur.

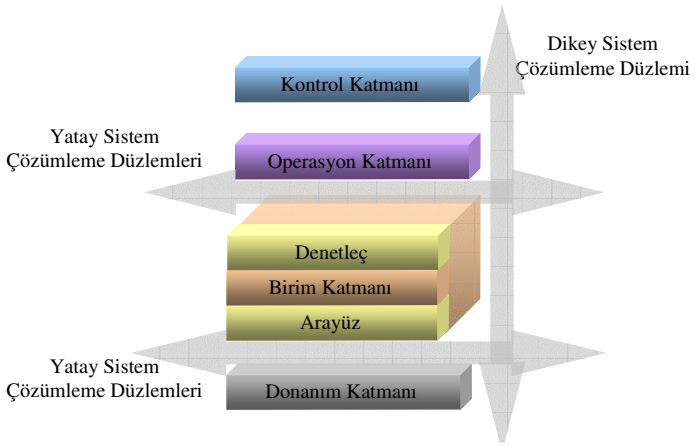
Atış Kontrol Bilgisayarı işlemci kartları, mevcut 128MB bellek alanı ve genişleyebilir bellek alt yapısı ile genellikle gömülü sistemlerin aksine masaüstü bilgisayarlar kadar alt yapıya sahiptir. AKB Donanım mimarisi ve KMS çevre birimleri her ne kadar gömülü sistemler için hatırı sayılır bir alt yapı sunsa da; diğer gömülü sistemlerde olduğu gibi yazılım geliştirme aşamasında aşılması gereken genel problemler, alınması gereken önlemler vb. gömülü sistem mühendislik çalışmaları, sistem fonksiyonelliği ve performansı açısından çok önemlidir. Genellikle yazılım çalışmaları başladığında hedef platformlar, sensörler, I/O kartları hazır değildir. Yukarıdaki açıklamalara benzer nedenlerle, aynı süreçlerin KMS Sistemi AKB yazılım çalışmalarında da yaşanacağı değerlendirilerek yazılım ön tasarımını takiben Yazılım Geliştirme Planı'na (YGP) uygun tasarım kararları işlenerek AKB Yazılım mimarisi için yerinde kararlar verilmiştir.

Örneğin, AKB içerisinde yer alan zamanlayıcıların, A/D çeviricilerin, KMS Sistemi'ndeki sensörlerin, eğleyicilerin simülatörlerini hazırlama ihtiyacı sistem geliştirme süreci değerlendirildiğinde açıkça görülmüştür. Yazılım geliştirme çalışmaları başladığında hazır olmayan birimlerin simülatörünü hazırlamak geniş zaman, yüksek uzmanlık ve farklı disiplinlerin bir arada çalışmasını gerektirir. Ayrıca, geliştirilen yazılım mimarisinin; konfigürasyon birimleri için hazırlanan simülatörlerin gerçek birimler için hazırlanacak yazılımlarla kullanılabilmesi, analiz aşamasında belirlenen donanım konfigürasyon birimlerindeki olası değişikliklere uyum gösterebilmesi ve iyileştirme çalışmalarının kolayca gerçekleştirilebilmesi için esnek bir alt yapı sunmasını gerektirmektedir.

Sistem ve yazılım geliştirme süreci, yukarıdaki örneklerin ışığında değerlendirildiğinde, analiz aşamasında sistem ve alt sistem isterlerinin optimum şekilde belirlenmesi, uzmanlık disiplinleri arasında ortak bir paydada koordinasyonun ve terminolojinin sağlanması, açısından önemlidir. Gerçek zamanlı ve gömülü sistemlerde sağlamlık, güvenilirlik, zamanlama, performans kısıtları, yazılım çalışmalarının gerçek donanım konfigürasyon birimleri ile başlama ihtimalinin azlığı ve yazılım konfigürasyon alt birimlerinin yazılımın genelini en az etkileyecek şekilde değiştirebilir olmasını sağlayacak bir mimari üzerine kurulması gereği gibi nedenler yazılım modelleme süreci ihtiyacını tariflemektedir.

KMS Projesi yazılım geliştirme süreci boyunca, sistem ve alt sistem gereksinimlerini girdi olarak alarak UML metodolojisine uygun modelleme süreci benimsenmiş ve sistem çözümleme / tasarım aşamasında farklı açılardan sistem değerlendirilerek modeller oluşturulmuştur. Bu modeller ile yazılım ekibinin sistemi değerlendirmesi, kavraması, proje ekibi ile görüş alışverişinde bulunması ve kritik konuları diğerlerinden ayırarak öncelikli gereklerin belirlenmesini, tasarım kararlarının alınması ve bu şekilde Yazılım Gereksinim Özellikleri (YGÖ) dokümanının olgunlaşması sağlanmıştır.

KMS Sistemi AKB uygulama yazılımları tasarım ve geliştirme çalışmaları, yazılım gerekleri yatay ve dikey düzlemlerde (Şekil 3) çözümlenerek ele alınmış ve bu katmanlı yapı UML CASE Aracı ile geliştirilen yazılımda doğrudan uygulanmıştır. Bu şekilde, UML modelleme süreci uygulanarak sistem tasarımı problem tabanına indirgenerek tasarım çıktılarının anlaşılabilirliği, yeniden-kullanılabilirliği, esnekliği ve taşınabilirliği sağlanmıştır.



Şekil 3. KMS Yatay ve Dikey Sistem Çözümleme Düzlemleri

UML metodolojisi ile sistem analizi ve tasarımı için uygulanacak modelleme sürecinde aşağıdaki diyagramlar kullanılabilir.

- Kullanıcı Senaryo Diyagramları (Use-case Diagrams)
- Sınıf Diyagramları (Class Diagrams)
- Nesne Diyagramları (Object Diagrams)
- Ardıl Etkileşim Diyagramları (Sequence Diagrams)
- İşbirliği Diyagramları (Collaboration Diagrams)
- Durum Diyagramları (State Diagrams)
- Etkinlik Diyagramları (Activity Diagrams)

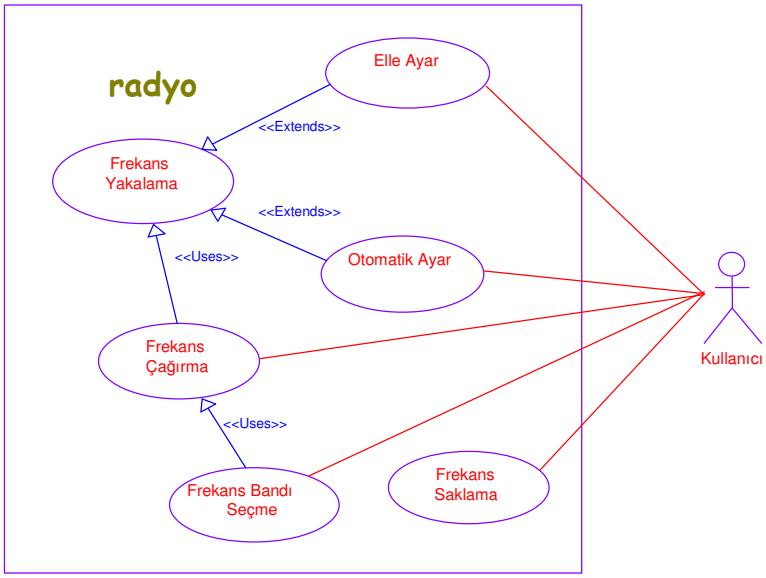
- Bileşen Diyagramları (Component Diagrams)
- Görev Atama Diyagramları (Deployment Diagrams)

KMS Sistemi AKB uygulama yazılımlarının analizi, tasarımı ve geliştirilmesi aşamasında yukarıda belirtilen diyagramlardan bir kaçı aşağıda alt başlıklarda anlatıldığı şekilde modelleme sürecinde yer almıştır.

3. Kullanıcı Senaryoları Modelleme

Kullanıcı Senaryoları Modelleme, sistem seviyesi fonksiyonların ve kısaca sistemin ne yapacağının tanımlandığı analiz yöntemidir, bu amaçla Kullanıcı Senaryo Diyagramları (Şekil 4) kullanılır. Kullanıcı Senaryo Diyagramları, sistemin, son kullanıcının bakış açısından analizinin yapılmasını ve gereksinimlerin belirlenmesini sağlar, tasarım yöntemi değildir. Kullanıcı Senaryo Diyagramları ile sistemin kabiliyetleri, sistemle arayüzü olan kullanıcılar belirlenerek sistem fonksiyonel gereklerinin, performans ve kalite isterlerinin (Quality of Service- QoS) netleşmesi sağlanır [2].
QoS:

- En-kötü durum uygulama senaryoları (Worst-case execution time)
- Ortalama uygulama senaryoları (Average execution time)
- Çıktı miktarı (Throughput)
- Tahmin edilebilirlik (Predictability)
- Kapasite (Capacity)
- Güvenlik (Safety)
- Güvenilirlik (Reliability)



Şekil 4. Örnek Kullanıcı Senaryo Diagramı

KMS Sistemi'nde kullanıcı senaryolarını tanımlarken aşağıdakilere benzer sorularla amaç-odaklı olarak sonuca ulaşılmaya çalışılmıştır.

- Sistemin ana amacı ne?

- Sistemin yerine getirmesi gereken birincil ve ikincil öncelikli görevleri ne?
- Sistemin nelerle arayüzü olacaktır?
- Kullanıcı senaryoları ne ile başlıyor?
- Sistemin aktörleri kimler?
- Aktörlerin, sistemden ve sistemin dışından beklentileri ne?
- En sık karşılaşılan senaryolar ne?
- Neler hataya yol açar?

Kullanıcı Senaryo Diyagramları'nın tanımlanmasında genel kurallar yoktur. Daha çok sistem mühendisliği koordinasyonunda uzmanlık disiplinlerinin tecrübe ve prensiplerine dayanır. Sistemin NE yapacağını tanımlar, NASIL yapacağı sorusu üzerinde durmaz [2].

4. Nesnelerin ve Katmanların Belirlenmesi

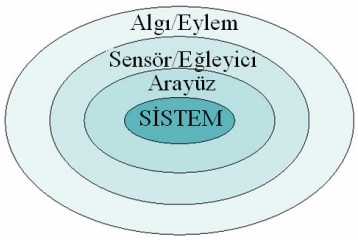
Kullanıcı senaryolarını modelleme çalışmalarının bir çıktısı da yazılımın sunduğu kabiliyetlerin ve verilerin kolayca sınıflandırılmasıdır. Kullanıcı senaryoları, yazılımı oluşturan yazılım nesnelerinin sınıflandırılmasına katkıda bulunur.

Nesne Nedir? Sistemle ilgili bir takım problemleri çözmek için tanımlanmış metot ve verileri barındıran; sistem, alt sistem ve birimleri temsil eden yürütüm süresi yazılım/sistem elemanlarıdır.

Nesneler aşağıdaki tiplerde olabilirler.

- Yazılım Elemanları: Bellekte belirli bir alana yerleşmiştir.
  Örn: HızDöngüsüDenetleci
- Elektronik Donanım Birimleri: Sistemde belirli bir fiziksel alana yerleşmiştir.
  Örn: AçısalKonumAlgılayıcı
- Mekanik Donanım Birimleri: Sistemde belirli bir fiziksel alana yerleşmiştir.
  Örn: HidrolikMotor
- Sistem Birimleri: Belirli bir fiziksel alana yerleşmiştir.
  Örn: Servo Alt Sistemi

KMS Sistemi'ni oluşturan nesneler, Şekil 3'de verilen sistem çözümleme yöntemine uygun olarak soğan halkası benzeşim (Onion Skin Analogy) modeli (Şekil 5) çerçevesinde belirlenmiş ve sınıflandırılmıştır [1].



Şekil 5. Soğan Halkası Benzeşim Modeli

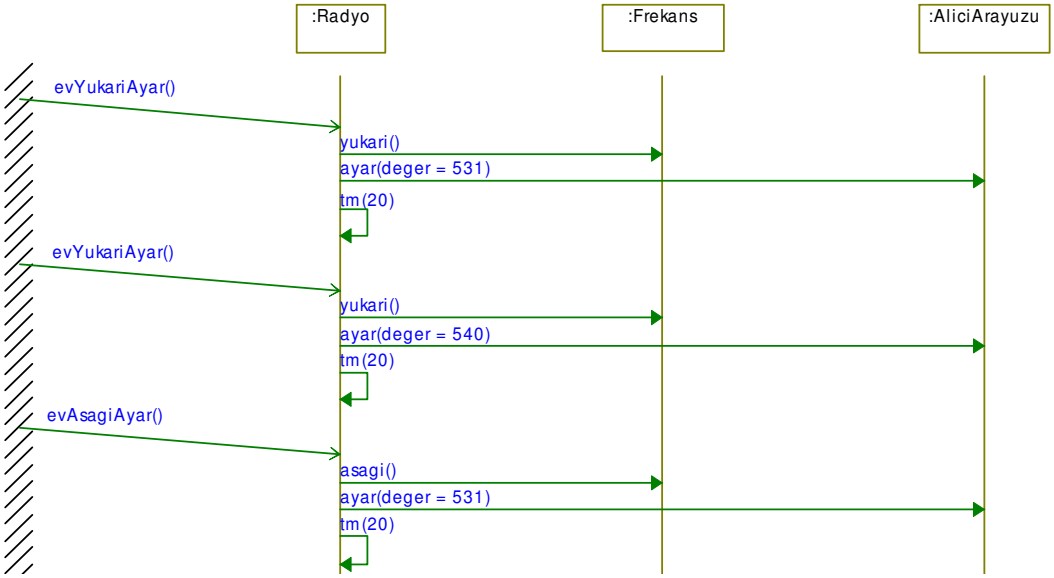
Halkaların tanımları aşağıdaki şekildedir.

Algı/Eylem: Kullanıcı Senaryoları
Sensör/Eğleyici: Giriş/Çıkış
Arayüz: Fiziksel Bağlantı Birimleri

KMS Sistemi uygulama yazılımları için uygulanacak nesnelerin kullanıcı senaryoları, benzeşim modelleri, problem çözümleme düzlemleri kullanılarak sınıflandırılmasını takiben kullanıcı senaryoları etkileşim ve durum modelleri ile detaylandırılmıştır.

5. Nesneler Arasındaki Etkileşimlerin Modellenmesi

Sınıflandırılmış, birbiri ile ilişkili olan nesne gruplarının ortak karakterlerini tanımlamak, birbirleri arasındaki tipik ve kritik ilişkileri yakalamak, nesneler arası işbirliği kabiliyetlerini tanımlamak ve test etmek için gerçek zamanlı sistemlerde yaygın olarak ardıl etkileşim diyagramları (Şekil 6) kullanılır [1, 2]. Bu nesne grupları nesneler, kullanıcı senaryo nesneleri, sistem, alt sistem ve aktörler olabilir.
KMS Sistemi uygulama yazılım nesne grupları arasındaki etkileşim modelleri, kullanıcı senaryolarının nasıl yapılacağını tanımlamak için hazırlanmıştır.

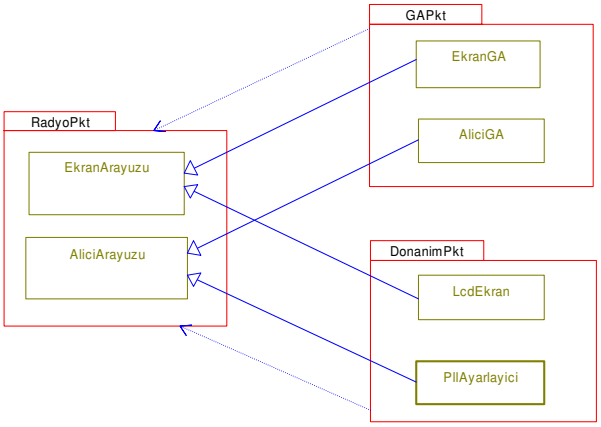


Şekil 6. Örnek Ardıl Etkileşim Diyagramı

6. Nesnelerin Durağan Karakterlerinin Modellenmesi

Sınıf metotları, ilişkileri ortak ve aynı anlamı paylaşan nesneler topluluğunun ortak tanımıdır. Nesnelerin işlevlerini nasıl yerine getireceklerini ve hangi verileri içereceklerini tanımlar. Sınıflar yazılımın durağan yapısının tanımlanmasında kullanılır. KMS Sistemi AKB uygulama yazılımında nesneler arasındaki ilişkiler görsel olarak sınıf diyagramları ile tanımlanmıştır ve diyagramlar karmaşık ilişkilere sahip olmayan nesne gruplarından oluşsa bile sınıf diyagramları ile sınıf ilişkileri modellenmiştir.
İlişkiler yürütüm süresi boyunca kullanılacak nesnelerin ve bu nesneler arasındaki haberleşmenin nasıl olacağını tanımlar (Şekil 7). Nesneler, ilişkiler sayesinde birbirlerinin işlevlerini ve verilerini kullanırlar. KMS Sistemi AKB uygulama yazılımlarında katmanlı yapının uygulanması esnasında 3 farklı tip ilişki kullanılmıştır.

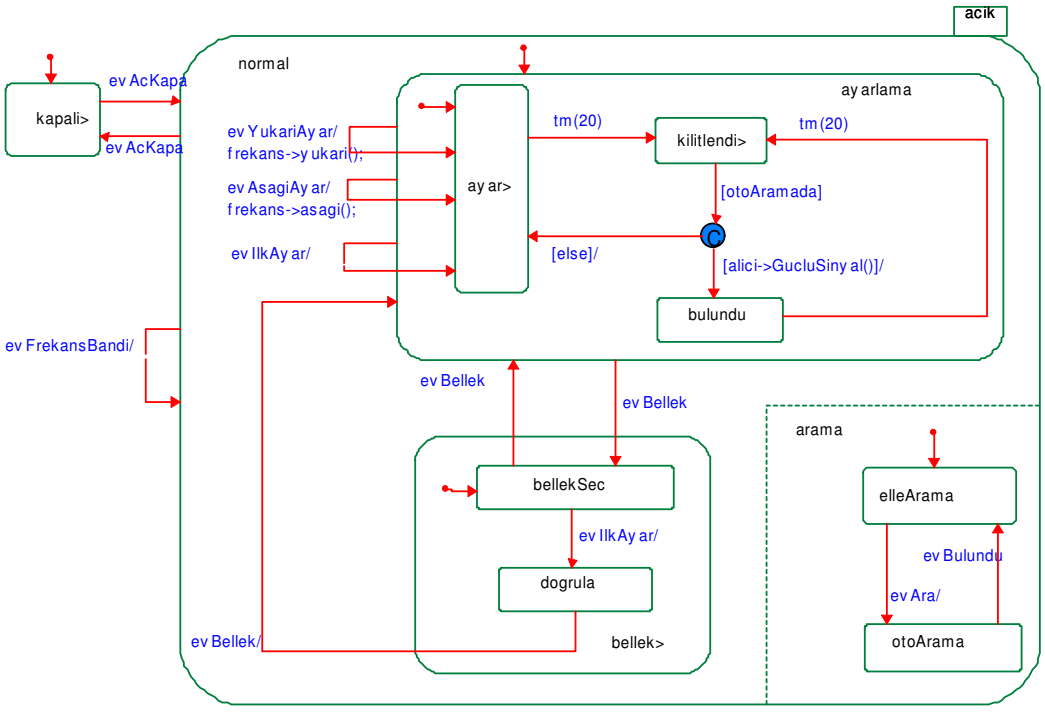
- Bağıntı İlişkisi (Association)
- Genelleme İlişkisi (Generalization)
- Bağımlılık İlişkisi (Dependency)



Şekil 6. Örnek Nesne Modelleme Diyagramı

7. Nesnelerin Devingen Karakterlerinin Modellenmesi

Nesnelerin başlangıç durumlarını, yürütüm süresi boyunca durumunun nasıl değiştiğini ve bitiş durumlarını modellemek için durum diyagramları kullanılır. Gerçek zamanlı sistemlerde nesnenin davranışı kendisine gönderilen iletilerin yanı sıra o an içinde bulunduğu duruma göre farklılık gösteriyorsa durum diyagramları kullanılır (Şekil 8). Diğer nesneler ile koşut zamanlılığı denetimi ve zaman iletilerinin kullanılmadığı durumlarda doğrudan zamanla ilişkilendirilmezler. KMS Sistemi AKB uygulama yazılımlarında durum diyagramları idamesi ve anlaşılabilirliği zor olan nesneler için hazırlanmıştır. KMS Projesi'nde yazılım geliştirme aracı olarak kullanılan UML CASE Aracı durum diyagramları, iç içe durumlar ve eş zamanlı durumlar tanımlanması, durum koşullarının tanımlanması, durumların içinde, giriş ve çıkışındaki eylemlerin tanımlanması, alt durumların geçmiş durum bilgilerini kullanabilmesi gibi özellikler sunmaktadır.



Şekil 7. Örnek Durum Diyagramı

8. Sonuç

KMS Sistemi AKB uygulama yazılımları, UML modelleme süreçleri tabanlı olarak analiz edilmiş, tasarımları yapılmış ve geliştirilmiştir. Tasarım ve geliştirme çalışmaları, yazılım gerekleri yatay ve dikey düzlemlerde çözümlenerek ele alınmış ve bu katmanlı yapı UML CASE Aracı ile geliştirilen yazılımlarda doğrudan uygulanmıştır. Bu şekilde, sistem tasarımı problem tabanına indirgenerek tasarım çıktılarının anlaşılabilirliği, yeniden-kullanılabilirliği, esnekliği ve taşınabilirliği sağlanmıştır.

Bu bildiride, katı gerçek zamanlı bir sistem olarak tanımlanan KMS Sistemi'nin; zamanlama mekanizmalarının, güvenlik ve güvenilirlik için alınan önlemlerinin, işlemciler arası haberleşmeyi hızlandıran yazılım mimarilerinin ve tasarım şablonlarının detaylarına girilmemiştir. Genel olarak UML modelleme sürecinin KMS Sistemi AKB uygulama yazılımlarında nasıl uygulandığı anlatılmıştır.

KMS Sistemi sistem geliştirme sürecinde açıkça görülmüştür ki karmaşık gerçek zamanlı sistemlerde modele dayalı yazılım süreçlerinin ve araçlarının kullanılması yazılımın anlaşılabilirliği, yeniden kullanılabilirliği, esnekliği, taşınabilirliği ve gerçek zamanlı sistemlerin tanımlı kısıtlamalarının sağlanması açısından en doğru kararlardan biridir.

9. Kaynakça

[1]. Douglass, B. P. , "Real Time UML: Advances in The UML for Real-Time Systems" 3. Baskı, Şubat 2004.

[2]. Douglass, B. P. , "Real-Time UML", Embedded Systems Conference 2004.