JACK Kanı-İstek-Hedef Modeline dayalı Çok-etmenli bir Kütüphane Yönetim Sisteminin Prometheus Metodolojisi ile Geliştirilmesi

Yağız Kaymak[1] Kemal Deniz Teket[2] Sercan Demirci[3] Sinem Getir[4] Rıza Cenk Erdur[5] Geylani Kardaş[6]

[1,2,3,4,6]Uluslararası Bilgisayar Enstitüsü, Ege Üniversitesi, İzmir
[5]Bilgisayar Mühendisliği Bölümü, Ege Üniversitesi, İzmir
[1]e-posta:yagiz.kaymak@ege.edu.tr [2]e-posta:denizkemal@gmail.com [3]e-posta:sercan.demirci@ege.edu.tr
[4]e-posta:sinem.getir@ege.edu.tr [5]e-posta: cenk.erdur@ege.edu.tr [6]e-posta:geylani.kardas@ege.edu.tr

Özetçe

Günümüzde birçok karmaşık sistemin tasarım ve uygulamasında özerk, karşıt-eylemli ve hedef-yönelimli akıllı etmen sistemlerinin kullanımı giderek yaygınlaşmaktadır. Ancak yapılan çalışmalar incelendiğinde, etmen sistemlerinin bu yapılarından dolayı tasarımları ve gerçek uygulaması arasında bir boşluk olduğu görülmektedir. Bu noktadan yola çıkarak, bu çalışmada Prometheus etmen tabanlı yazılım geliştirme metodolojisi uygulanarak JACK Kanı-İstek-Hedef modeline dayalı bir çok-etmenli sistemin tasarım ve uygulaması gerçekleştirilmiştir. Geliştirilen sistem çok etmenli bir kütüphane yönetim sistemidir. Seçilen örnek senaryo yardımıyla çok etmenli bir sistemin, bir metodoloji takip edilerek ve etmen tabanlı bir yazılım geliştirme ortamı kullanılarak nasıl gerçekleneceği tüm aşamalarıyla anlatılmıştır. Çalışmada, Prometheus metodolojisinin ve JACK çerçevesinin gözlemlenen avantaj ve dezavantajları açıklanmış, metodoloji, geliştirme ortamının birbirleri ile uyumluluğu aktarılmış ve elde edilen deneyimler paylaşılmıştır.

1. Giriş

Yazılım etmenleri ("*software agents"*), belirli bir ortamda konumlanmış ve tasarım hedeflerini uygulamak için bu ortam üzerinde özerk eylem gerçekleştirebilme yeteneğine sahip olan bilgisayar sistemleridir [1]. Bir etmen, bulunduğu ortam ile ilgili genellikle kısıtlı bilgiye sahiptir. Bu nedenle üzerinde çalışılan ortam karmaşıklaştıkça veya genişledikçe, birbirleriyle iletişim kurup, etkileşime giren birden fazla etmene ihtiyaç duyulmaktadır. Bu şekilde, birbirleriyle etkileşim halinde olan birçok etmenin bulunduğu sistemlere çok-etmenli sistemler (ÇES) adı verilmektedir.

ÇES, günümüzde var olan yazılımlardan farklı olarak akıllılık, özerklik ve sosyallik gibi özelliklere sahiptir. Etmenler, çoğunlukla insan müdahalesi olmadan kullanıcıları adına hareket ederek kullanıcılarının hedeflerini gerçekleştirmek amacıyla çalışırlar. Bu doğrultuda, gerektiğinde birbirleriyle haberleşerek ve etkileşime girerek ortak çalışmanın gerekliliklerini yerine getirirler.

Yazılım etmenlerinin sıkça kullanılmaya başlamasıyla etmen tabanlı yazılım mühendisliği (ETYM) daha sık gündeme gelmeye başlamıştır. ETYM'nin esas amacı, geliştirmesi ve bakımı pahalı olmayan etmen tabanlı metodoloji ve yazılım araçları oluşturmaktır [2].

Etmen tabanlı yazılımlarda metodoloji, yazılım geliştirme sürecinde kullanılan kavramların kümesi, modelleme aşamasındaki notasyonlar (gereksinimler, tasarım ve uygulama) ve yazılım geliştirmek için izlenen süreçten oluşmaktadır [3]. Böyle sistemlerin geliştirilmesini sağlayan ve farklı amaçlara hizmet edebilen birçok metodoloji bulunmaktadır.

Bu çalışmada, yukarıda sözü edilen ETYM metodolojilerinden biri olan Prometheus metodolojisi izlenerek, Kanı-İstek-Hedef (KİH) ("*Belief-Desire-Intention" (BDI)*) modeline uygun olarak davranış gösteren etmenleri içeren bir çok-etmenli kütüphane yönetim sistemi geliştirilmiştir. Tasarlanan sistemin gerçekleştirilmesinde popüler etmen platformlarından biri olan JACK kullanılmıştır.

Bir KİH [4] modeline göre hareket eden bir etmen hangi amaçlara nasıl erişebileceği hakkında karar vermektedir. *Kanı*lar etmenin çevresi hakkında bildiklerini temsil etmektedir. *İstek*ler ise etmenin elde etmek istediklerine karşılık gelmektedir. Son olarak, etmenin düşünülmüş davranışları olarak da ifade edebileceğimiz *hedefler*, etmen amaçlarının elde edilmesi için işletilen etmen planlama mekanizmasını içermektedir. Tanıttığımız sistem içerisinde yazılım etmenleri bu modele uygun olarak kullanıcıları adına kitap alımı, kitap iadesi, kitap rezervasyonu gibi görevleri yerine getirmektedirler.

Bu çalışma kapsamında JACK[1] çerçevesi ile uyumluluğu bilinen Prometheus metodolojisi [5] kullanılmıştır. Prometheus metodolojisi, analiz, mimari tasarım ve detaylandırılmış tasarım aşamalarını içeren yinelemeli bir yazılım mühendisliği sürecidir ve amacı akıllı etmenler (özellikle KİH etmenleri) geliştirmektir.

Farklı etmen mimarilerine ve ihtiyaçlarına göre JADE [6], JADEX [7, 8] ve JACK [9] gibi çeşitli etmen tabanlı yazılım geliştirme ortamları bulunmaktadır. Bu çalışma kapsamında kullanılan yazılım geliştirme ortamı ise JACK'dir.

JACK muadili birçok çerçeveden ("framework") farklı olarak görsel modelleme ve tasarıma imkan veren bir araç desteğini sunmaktadır. Ayrıca, etmenler, planlar, inanç kümeleri, olaylar ve yetenekler gibi bileşenleri gruplandırarak farklı bileşenleri ilişkilendirmeye olanak tanımaktadır. JADEX'te ise böyle bir özellik bulunmamaktadır. Öte yandan, JACK, ticari bir çerçeve olduğundan, düzenli güncellemeler ve yeterli miktarda dokümantasyon sağlamaktadır. JADEX'in son sürümünde bulunan hatalar ve JACK'e kıyasla daha az kullanışlı dokümantasyona sahip oluşu, JADEX yerine JACK'in tercih edilmesinin nedenleri arasında verilebilir. JADE ise KİH modelini desteklemediği için bu çalışmanın gerçeklenmesi için tercih edilmemiştir.

[1] http://www.aosgrp.com/products/jack/index.html, son erişim tarihi 27.02.2012.

JACK, 1997 yılında Avustralya Yapay Zeka Enstitüsü'nün eski üyeleri tarafından oluşturulan ve KİH modelini temel alan bir etmen platformudur. Günümüzde JACK, savunma sistemlerinden endüstriye kadar birçok uygulama alanında kullanılmaktadır. Bu uygulamalara ait detaylı bilgiler [10]'da bulunabilir.

Bildiride, gerçek uygulamalarda kullanılan JACK çerçevesi ile etmen tabanlı bir kütüphane yönetim sisteminin tasarım ve uygulaması adımları aktarılmakta; takip edilen Prometheus metodolojisi ve JACK çerçevesi ile bir ÇES'in nasıl gerçeklenebileceği gösterilmektedir. Bunun yanı sıra, çalışmada Prometheus ile oluşturulan sistem tasarımını JACK'e aktarırken karşılaşılan zorluklar dile getirilmiş ve Prometheus metodolojisi ile JACK ortamının ne gibi avantaj ve dezavantajlar barındırdığına değinilmiştir. Ayrıca, uygulama geliştirme sürecinde elde edilen deneyimler aktarılmıştır.

Bildirinin geri kalan kısmı şu bölümlerden oluşmaktadır: Bölüm 2'de konu ile ilgili diğer çalışmalar özetlenmiştir. Bölüm 3'te, Prometheus metodolojisi kısaca açıklanmış ve örnek kütüphane yönetim sisteminin Prometheus ile analiz ve tasarımı detaylandırılmıştır. Bölüm 4' te JACK etmen tabanlı yazılım geliştirme ortamı ile ilgili genel bilgiler verilmiş ve örnek kütüphane yönetim sisteminin JACK ile gerçeklenmesi anlatılmıştır. Bölüm 5'te elde edilen deneyimler, çalışmanın değerlendirilmesi ve sonuçlar bulunmaktadır.

2. İlgili Çalışmalar

ÇES tasarımı ve analizi konusunda literatürde çeşitli metodolojiler bulunmaktadır [11]. Bu metodolojilerin büyük bir kısmı nesne tabanlı veya bilgi tabanlı olmak üzere ikiye ayrılmıştır. Birçok modelin sözdizimi, birleşik modelleme dilinden alınarak oluşturulmuş olsa dahi etmen tabanlı sistemlerde metodoloji kavramı, nesne tabanlı yaklaşımdan farklılıklar göstermektedir [12]. Çalışmalar incelendiğinde en çok kullanılan metodolojilerin Gaia [13], Tropos [14] ve Prometheus [5] olduğu dikkat çekmektedir.

ÇES kullanılarak gerçekleştirilmiş bir kütüphane yönetim sistemi [15]'te karşımıza çıkmaktadır. Jain ve Dahiya, [15]'te Gaia metodolojisini takip ederek sistem tasarımını gerçekleştirmişlerdir. Bahsi geçen çalışmadaki sistem tasarımı, geliştirdiğimiz uygulamanın aksine KİH mimarisini takip etmemektedir. Ayrıca, çalışmada detaylı tasarım aşamasında bilgi tabanı kullanımının eksik olduğu belirtilmektedir.

Prometheus metodolojisinin izlendiği bir çalışma olan [16]'da tahliye planlaması üzerine gerçekleşen bir simülasyon Prometheus aşamaları ile tasarlanmıştır. Diğer yandan Prometheus metodolojisi ile JACK etmen tabanlı yazılım geliştirme ortamının birlikte kullanıldığı birçok çalışma literatürde karşımıza çıkmaktadır. Bu çalışmalardan çoğunluğu askeri ve endüstriyel alanlarda kullanılmaktadır. Bahsi geçen çalışmalardan bazıları şu şekilde özetlenebilir: Lucas ve ark. [17]'de uzaktan kumanda ile yönetilen insansız hava taşıtı sistemini ÇES kullanarak uygulamışlardır. [18]'deki çalışmada ise farklı savaş alanları, ÇES ile simüle edilerek, savaş alanlarındaki lojistik kararların plan yönetim sistemi aracılığıyla nasıl belirleneceği aktarılmıştır. Marc ve ark., [19]'da askeri kaynakların (özellikle uçakların) nasıl konuşlandırılacağına ÇES ile karar verecek bir uygulama geliştirmişlerdir. [20, 21]'de Avustralya Meteoroloji Bürosu için hava durumu alarm sistemi uygulamasının nasıl gerçeklendiği anlatılmıştır. [22]'de ise insan hareketlerini takip ve tespit eden mobil robotların ÇES ile uygulanışı aktarılmıştır.

Literatürde, KİH mimarisiyle gerçekleştirilmiş etmen tabanlı bir kütüphane yönetim sistemi bulunmamaktadır. Ayrıca iş alanı bağımsız olarak tüm bu çalışmalar değerlendirildiğinde özellikle Prometheus'a dayalı veya JACK KİH modeli kullanılarak gerçekleştirilen çalışmalarda etmen yazılımı geliştiricilerinin ihtiyaç duyacağı deneyimlerin az miktarda yansıtıldığı gözlenmiştir. Çalışmamızın söz konusu uluslararası ETYM çalışmalarına bu yönde bir katkı yaptığı düşünülmektedir. Ülkemizde ise etmen tabanlı yazılım sistemlerinin geliştirilmesine yönelik çalışmalar günümüzde az sayıdadır ve bunların pek azı ulusal literatürümüzde yer almıştır. Örnek olarak [23]'te, bir konferans yönetim sistemi geliştirilmesi, Gaia metodolojisi kullanılarak adım adım açıklanmaktadır ancak detaylı bir uygulama verilmemiştir. [24]'te ise örnek bir öğrenci bilgi sistemi, KİH mimarisini desteklemeyen JADE ile uygulanmıştır. Ulusal boyuttaki bu çalışmalarda da, JACK tabanlı veya KİH modeline dayanan etmen yazılım sistemlerinin geliştirilmesinin göz önüne alınmadığı görülmektedir. Bu açıdan da çalışmamızın ulusal boyuttaki bu eksikliği kapamaya yarayacak öncü çalışmalardan biri olduğuna inanılmaktadır.

3. Prometheus Metodolojisi ile Sistem Tasarımı

Prometheus, problem tanımlama, tasarım, uygulama ve test/hata ayıklama için detaylı işlem süreci tanımlayan bir metodolojidir. Detaylı gelişim süreçlerine ek olarak yazılım geliştirme boyunca bir dizi veri ("*artifact*") elde edilir. Bu verilerden bazıları ilerleyen zamanlarda kullanılmak için saklanırken bazıları ise sadece geliştirme esnasındaki ara aşamalarda kullanılır [3].

Prometheus metodolojisini diğer metodolojilerden ayıran özellikler ise şu şekilde ortaya koyulabilir [3]:

Gaia, Prometheus gibi uzun yıllar içerisinde birçok kişi tarafından geliştirilmiş bir metodolojidir. Ancak Gaia, detaylandırılmış tasarım sürecine yeterli destek sağlamamaktadır. Bunun yanında sistem tanımlama ve mimari tasarım aşamalarının Prometheus'a benzediği söylenebilir.

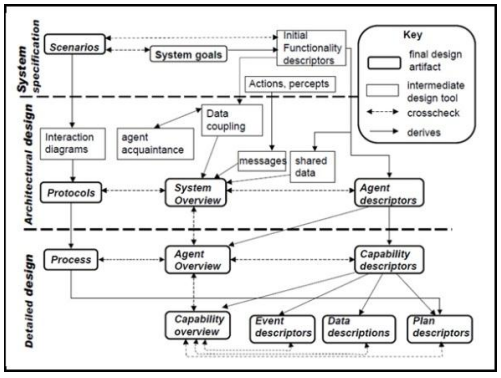
Tropos'a göre Prometheus, özellikle mimari tasarım aşamasında daha detaylı bir sürece sahiptir. Ancak Prometheus'ta, Tropos'ta bulunan ön gereksinimler ("*early requirements*") aşaması bulunmamaktadır. Ayrıca Prometheus, araç desteği ve çapraz kontrol mekanizmalarına sahipken, Tropos'un araç desteği diyagram editöründeki basit bir formdan ibarettir.

MaSE [25], güçlü araç desteği olan az sayıdaki metodolojilerden biridir. Ancak MaSE karşıt-eylemli ve hedef-yönelimli davranışlar içeren plan tabanlı etmenlerin oluşturulmasına destek sağlamamaktadır.

JACK platformuna uyumlu bir metodoloji olarak seçilen Prometheus, temel olarak üç aşamadan oluşur [3]:

- *Sistem tanımlamaları:* Hedefler, sistemin temel işlevsellikleri, senaryolar, algılar ve eylemler tanımlanır.
- *Mimari tasarım:* Sistem tanımlama aşamasında elde edilen verilerden yararlanarak etmen tipleri ve etmenler arası etkileşimler ortaya konulur.
- *Detaylandırılmış tasarım:* Etmenlerin iç yapısı tasarlanır ve her etmenin görevini nasıl tamamlayacağı belirlenir.

Bu aşamalar Şekil 1'de ayrıntılı olarak gösterilmektedir (Anlam kayıplarını engellemek için bildirinin tamamındaki tüm şekillerin içeriği İngilizce olarak verilmiştir.)



Şekil 1: Prometheus Metodolojisinde Tasarım Sürecindeki Aşamalar [3].

Prometheus metodolojisi, sistem tasarımını kolaylaştıran *Prometheus Design Tool*[2] (PDT) adında bir araç desteğine sahiptir. PDT, Prometheus metodolojisi ile etmen tabanlı sistem geliştirmeyi sağlayan bir araçtır. Bu çalışmanın temel aldığı ve çalışma kapsamında kullandığımız ve genişlettiğimiz bir örnek kütüphane yönetim sisteminin tasarımı [26], Prometheus metodolojisi takip edilerek ve PDT aracı kullanılarak oluşturulmuştur. Aşağıdaki altbölümlerde bahsedilen aşamaların detayları ve kütüphane sistemi için nasıl tasarlandığı aktarılmaktadır.

3.1. Sistem Tanımlamaları

Bu aşamada geliştirilmesi hedeflenen sistem için aşağıdaki adımlar uygulanır:

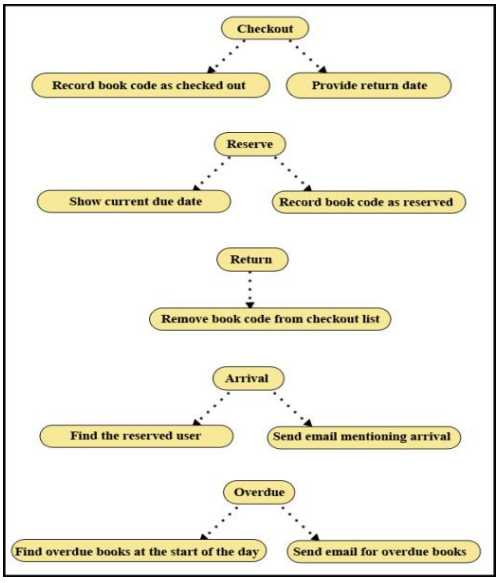
- Hedeflerin Tanımlanması
- Senaryoların Belirlenmesi
- İşlevselliklerin Belirlenmesi
- Algılar ve Eylemlerin Belirlenmesi

Çalışma kapsamında oluşturulan örnek kütüphane sisteminde Şekil 2'de gösterildiği gibi şu ana ve alt hedefler mevcuttur:

- Kitap alımı
  - Kitap kodunu, kullanıcı kimliğiyle birlikte teslim edilen kitaplar listesine kaydetmek
  - Teslim edilen kitap için geri dönüş tarihi oluşturmak
- Kitap iadesi
  - Kitap kodunu ve ilgili kullanıcı kimliğini teslim edilen kitaplar listesinden kaldırmak
- Kütüphanede bulunmayan kitapların rezervasyonu
  - Rezerve edilmek istenen kitap kodunu kullanıcı kimliği ile rezervasyon listesine kaydetmek
  - Rezerve edilen kitap için, teslim listesinde bulunan geri dönüş tarihini elde etmek
- Rezerve edilmiş kitapların iadesinde kullanıcıların bilgilendirilmesi
  - İade edilen kitapla ilgili kullanıcıyı rezervasyon listesinden bulmak
  - İlgili kullanıcıya rezerve ettiği kitabın geri dönüşüne ilişkin e-posta göndermek
- Teslim tarihi geçmiş kitaplar için kullanıcıların bilgilendirilmesi
  - Her gün başlangıcında kitap kayıtlarına ulaşmak
  - Teslim tarihi geçmiş kitaplar için kullanıcılara e-posta göndermek

Örnek kütüphane yönetim sisteminde yer alan senaryolar şunlardır:

- Bir kullanıcı kitap talep ettiğinde; *Request for checkout* (Kitap alım isteği) olayı tetiklenir ve *Checkout books* (Kitap alınması) işlevi gerçekleşir. Bunun üzerine *Provide book* (Kitap temin et) eylemi gerçekleştirilir ve *Checkout* (Alım) hedefine ulaşılmış olur.
- Kullanıcı bir kitap iade ettiğinde; *Book returned* (Kitap geri getirildi) olayı tetiklenir, *Return books* (Kitapların iade edilmesi) işlevi gerçekleşir ve *Return* (İade) hedefine ulaşılmış olur.
- Kullanıcı kitap rezerve etmek istediğinde; *Request for reservation* (Rezervasyon isteği) olayı tetiklenir, *Reserved book* (Kitap rezerve edilmesi) işlevi gerçekleşir. *Provide current due date* (Günün tarihinin temin edilmesi) eylemi gerçekleştirilir ve *Reserve* (Rezerve) hedefine ulaşılmış olur.
- Bir kitap kütüphaneye geri döndüğünde; *Reserve book arrives* (Rezerve edilmiş kitabın iadesi) olayı tetiklenir, *Arrival notification* (Kitap iade bildirimi) işlevi gerçekleşir, *Send arrival e-mail* (Kitap iade bildiriminde e-posta gönderimi) eylemi gerçekleştirilir ve *Arrival* (Kitap iade) hedefine ulaşılmış olur.
- Her gün başlangıcında; *Start of the day* (Gün başlangıcı) olayı tetiklenir, *Find overdue book* (Gecikmiş kitapların bulunması) işlevi gerçekleşir ve *Send overdue e-mail* (Gecikmiş kitaplar için e-posta gönder) eylemi gerçekleştirilir. Sonuçta *Overdue* (Gecikme) hedefine ulaşılmış olur.



Şekil 2: Kütüphane Yönetim Sistemi Hedef Diyagramı [26]

Çalışma kapsamında oluşturulan örnek kütüphane sisteminde şu işlevsellikler mevcuttur:

- Kütüphaneden kitap alımı ve kullanıcıya kitabın dönüş tarihinin bildirilmesi,
- Alınan kitapların kütüphaneye iadesi,
- Kütüphanede olmayan kitaplar için rezervasyon yapılması,

[2] http://www.cs.rmit.edu.au/agents/pdt, son erişim tarihi 27.02.2012.

- Rezerve edilmiş kitaplar için, kitapların iadesi esnasında kullanıcıların bilgilendirilmesi ve
- Teslim tarihi geçmiş kitaplar için kullanıcıların bilgilendirilmesi.

Bu aşamada, son olarak örnek kütüphane yönetim sisteminin algıları ve bu algılar sonucu gerçekleştireceği eylemler belirlenir:

- *Request for checkout* algısı sonucu *Checkout books* işlevi gerçekleştirilir. Bu işlevsellik dahilinde *Provide book* eylemi yapılır.
- *Book returned* algısı sonucu *Return books* işlevi gerçekleştirilir.
- *Start of day* algısı sonucu *Overdue books* işlevi gerçekleştirilir ve bu işlevsellik dahilinde *Send overdue email* eylemi yapılır.
- *Request for reservation* algısı sonucu *Reserve books* işlevi gerçekleştirilerek *Provide current due date* eylemi yapılır.
- *Reserved book arrives* algısı sonucu *Arrival notification* işlevi gerçekleştirilerek, *Send arrival email* eylemi yapılır.

3.2. Mimari Tasarım

Metodolojinin bu aşamasında planlar ile veri eşleşmeleri gerçekleştirilir. Ayrıca etmen tipleri belirlenerek planlar uygun etmenlere atanır. Sistem tanımlamalarında elde edilen veriler ile beraber bu aşamanın sonunda sistem genel bakış diyagramı oluşturulur. Mimari tasarım aşamasında takip edilecek adımlar aşağıdaki gibidir:

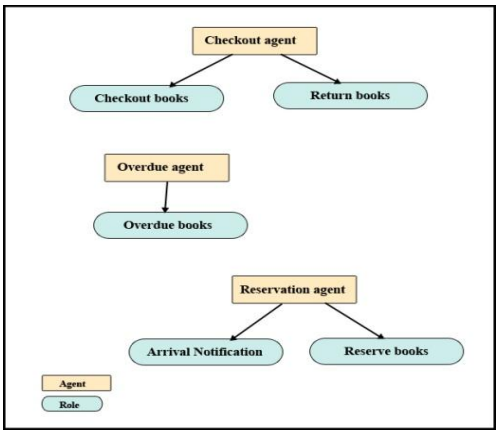
- Etmen Tiplerinin Belirlenmesi
- Etmenler Arası Etkileşimin, Etkileşim Diyagramı ve Protokolleri ile Gösterilmesi
- Sistem Yapısının Tasarlanması

Prometheus'un mimari tasarım aşamasını uygularken ilk adımda örnek kütüphane sistemimizde kullanılacak olan etmen tiplerini belirledik. Bu etmen tipleri: *Checkout agent* (Kitap temin etmeni), *Overdue agent* (Süresi geçmiş kitap kontrol etmeni) ve *Reservation agent* (Rezervasyon etmeni)'dir.

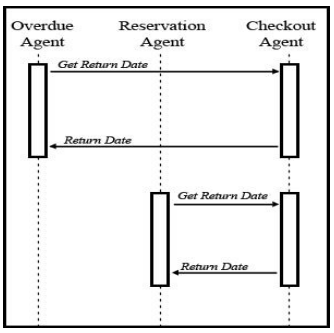
Checkout agent; herhangi bir kullanıcı, kütüphaneden kitap almak istediğinde kullanıcıya kitabı temin eden, verilen kitaba ait veritabanına kayıt düşen ve kullanıcıya kitabı teslim eden etmendir. *Overdue agent*, kullanım süresi dolan kitapları tespit eden ve bu kitapları bulunduran kullanıcılara e-posta gönderen etmendir. *Reservation agent*, başka kullanıcılarda olan kitaplar için rezervasyon yapan ve rezerve edilmiş kitaplar için ilgili kitapların iadesinde rezervasyon sahiplerine e-posta gönderen etmendir. Sistemdeki tüm etmen tiplerinin ve etmenlere ait rollerin gösterildiği diyagram Şekil 3'te görülebilir.

Mimari tasarımın ikinci adımında, etmenler arası etkileşimler belirlenmiştir. Örnek kütüphane yönetim sisteminde temel olarak iki etkileşim mevcuttur. Bunlardan ilki, *Overdue agent* ile *Checkout agent* arasında; diğeri de *Reservation agent* ile *Checkout agent* arasındadır.

Overdue agent gün başlangıcında o gün kütüphaneye iade edilmesi gereken kitap listesini öğrenmek için *Checkout agent* ile haberleşir. *Reservation agent* ise her kitap rezervasyon isteğinde *Checkout agent* etmeni ile haberleşerek rezerve edilen kitabın dönüş tarihini temin eder. Sistemdeki etmenler arası haberleşmeyi gösteren sistem etkileşim diyagramı Şekil 4'te verilmektedir.

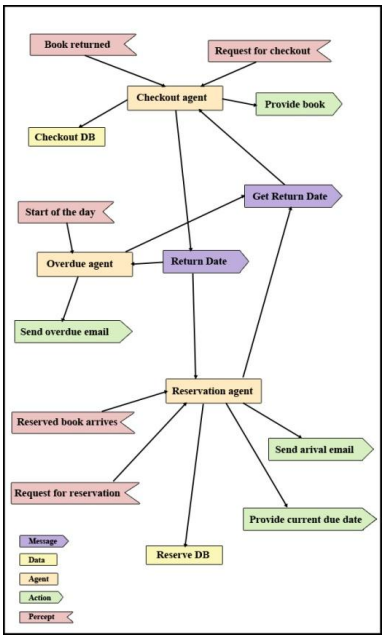


Şekil 3: Kütüphane Yönetim Sistemi Etmen Rol Diyagramı [26].



Şekil 4: Kütüphane Yönetim Sistemi Etkileşim Diyagramı.

Mimari tasarımın son aşamasında oluşturulan sistem yapısı Şekil 5'te görülmektedir.



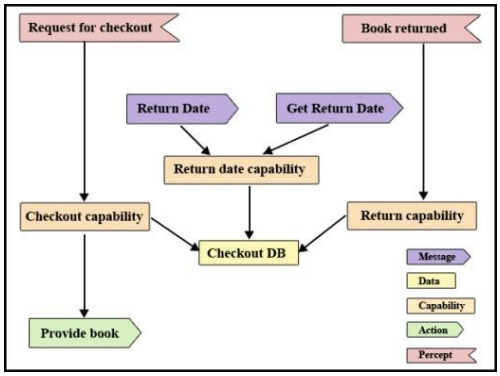
Şekil 5: Kütüphane Yönetim Sistemi Genel Bakış Diyagramı [26].

3.3. Detaylandırılmış Tasarım

Detaylandırılmış tasarımdaki adımlar şu şekilde özetlenebilir:

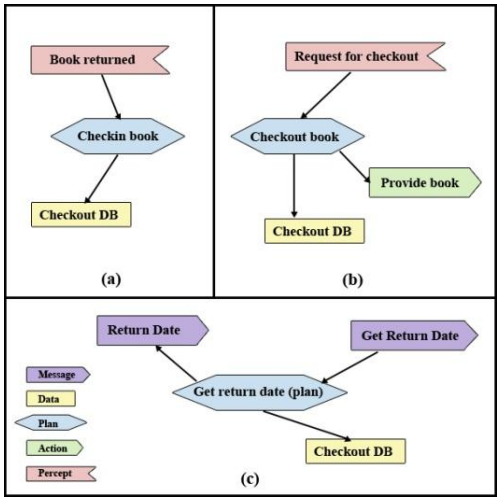
- Etmen yetenekleri genel bakış diyagramlarının hazırlanması
- Olayların ve mesajların oluşturulması
- Algı ve eylem detaylarının belirlenmesi

Örnek kütüphane yönetim sisteminde detaylandırılmış tasarım aşamasının ilk adımı olarak her etmenin genel bakış diyagramları oluşturulmuştur. Ancak yer kısıtı nedeniyle bu bildiride sadece *Checkout agent*'ın genel bakış diyagramı verilmiştir. Bu diyagram Şekil 6'da görülebilir.



Şekil 6: *Checkout Agent*'ın Genel Bakış Diyagramı [26].

Şekil 6'da görüldüğü gibi *Checkout agent*'ın üç adet yeteneği bulunmaktadır. Bu yetenekler, metodolojideki detaylandırılmış tasarımın son iki adımı gereği içerdikleri olayları, planları, algı ve eylemleri gösterecek şekilde oluşturulurlar. Bu detaylı gösterimler, yetenek genel bakış diyagramları olarak adlandırılır. *Checkout agent'*ın üç yeteneğine ait yeterlilik genel bakış diyagramları Şekil 7'de gösterilmiştir.



Şekil 7: Checkout agent için, *(a) Return capability, (b) Checkout capability, (c) Return date capability* yetenek genel bakış diyagramları [26].

4. Sistem Tasarımının JACK Etmen Tabanlı Yazılım Geliştirme Ortamı ile Gerçeklenmesi

JACK, KİH mimarisine dayanan bir çok etmenli sistem platformudur. Bu platformda, etmenlerin planları olaylar tarafından tetiklenir ve mesajlar da alt-olaylar olarak algılanır. Bir JACK programı şu bileşenlerden oluşur: etmenler, yetenekler, planlar, olaylar ve inanç kümeleri.

Bütün bileşenler başka bileşenlerle ilişkilidir. Örneğin bir etmen tanımlaması, hangi plan ve yetenekleri içerdiği, hangi inanç kümesine sahip olduğu, hangi mesajları aldığı ve gönderdiği gibi bilgilere sahiptir ve dolayısıyla diğer bileşenlerle ilişkilidir [3].

Çalışmamızda kullandığımız Prometheus metodolojisinin bazı kavramları, direkt olarak JACK ile ilişkilendirilememektedir. Örneğin bir Prometheus etmeni ve etmenin yetenekleri, JACK etmen ve yeteneklerine karşılık gelir. Ancak JACK platformunda Prometheus metodolojisinde kullanılan algı ve eylemlerin karşılığı bulunmamaktadır. Bunun yerine JACK'teki algılar, olaylara karşılık gelir; eylemler ise plan gövdesinin içerisine Java kodu olarak yazılır. Ayrıca KİH sistemlerinde (JACK dahil olmak üzere) hedefler direkt olarak gerçekleştirilemezler. JACK'teki hedefler, bu hedefe karşılık gelen bir olay tipi ile ifade edilir [3].

PDT'nin sağladığı gibi JACK de sistemin tasarlanması için görsel bir araç desteği sunmaktadır. Bu araç desteği sayesinde etmenler, planlar, olaylar, yetenekler ve inanç kümeleri görsel olarak oluşturulabilirler. JACK, görsel olarak oluşturulan her bileşen için kod iskeletlerini oluşturur.

PDT'nin ürettiği kodların JACK çerçevesine doğrudan aktarılmaması sebebiyle JACK'teki tasarım paleti kullanılarak sistemin genel bakış diyagramı modifiye edilmiştir. Şekil 8'de sistem tasarımının JACK'teki genel bakış diyagramı görülebilir.

Yer kısıtından dolayı genel bakış diyagramındaki tüm bileşenlerin açıklanması yerine detaylı anlatım için örnek olarak *ReservationAgent* etmeni seçilmiştir. *ReservationAgent*'ın iki yeteneği bulunmaktadır. Bunlar, *ReservedBookReturned ve RequestReturnDate*'dir. *ReservedBookReturned* yeteneği kapsamında, *ReservedBookReturn* olayı gerçekleştiğinde *HandleReservedBookReturn* planı tetiklenir. Bu planda, iade edilen kitap ile ilgili bir rezervasyon kaydının olması halinde, kitap için rezervasyon yapan kullanıcıya bilgilendirme e-postası gönderilir.

RequestReturnDate yeteneğinde ise, bir kullanıcıya tahsis edilmiş bir kitap için *CheckoutAgent* etmeninden kitabın geri dönüş tarihi istenir. Bunun için önce *ReturnDateRequest* olayı gerçekleşir. Bu olay, *SendReservationReturnDateRequest* planı ile ele alınır ve bu plan *ReturnDateRequestMessage* mesajını *CheckoutAgent*'a gönderir. *CheckoutAgent*'tan *ReturnDateReplyMessage* mesajı geldiğinde *ReservationAgent* bu olayı *HandleReservationReplyMessage* planı ile ele alır ve bu bilgileri *ReserveBS* inanç kümesine kaydeder.

Şekil 9'da *ReservationAgent* etmeninin genel bakış diyagramı ve yetenek diyagramları gösterilmiştir. Bildirinin geri kalan kısmında da *ReservationAgent* ve bu etmene ait yetenek, plan, olay ve inanç kümesine ait JACK kodları ve açıklamaları yer almaktadır.

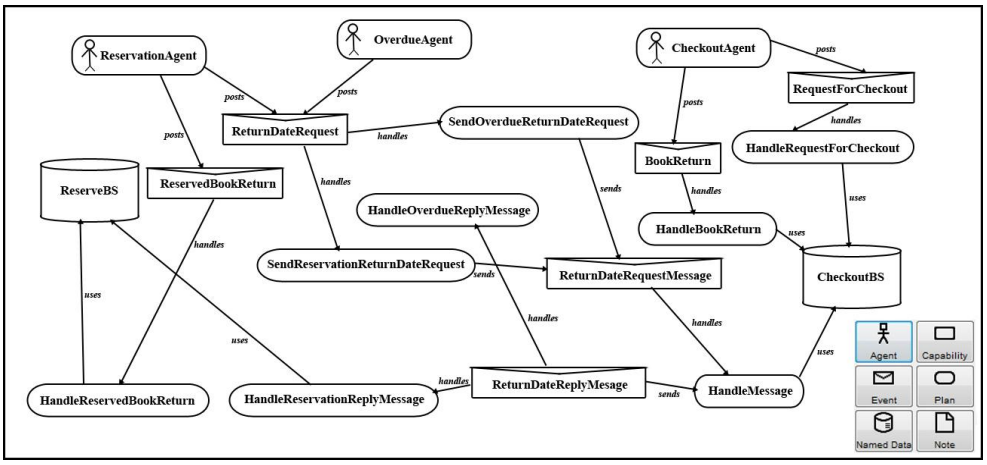
Listeleme 1'deki *ReservationAgent* etmen kod parçasında, etmenin sahip olduğu yetenekler *#has capability*, kendisine gönderdiği olaylar *#posts event* ve bütün etmen

üyelerinin kullandığı inanç kümeleri *#agent data* etiketleriyle tanımlanmıştır.

Listeleme 2'de, *ReservedBookReturn* yeteneğine ait kod parçası bulunmaktadır. Burada yetenek içerisinde, gelen bir mesajı karşılamak için *#handles external* etiketi ve yeteneğin barındırdığı planlar için *#uses plan* etiketi kullanılmaktadır.

```
1 public agent ReservationAgent extends Agent {
2  #has capability RequestReturnDate cap;
3  #has capability ReservedBookReturn cap1;
4  #posts event ReturnDateRequest ev;
5  #posts event ReservedBookReturn ev1;
6  #agent data ReserveBS ReserveBS();
7 }
```

Listeleme 1: ReservationAgent 'a ait JACK kod parçası.



Şekil 8: Tasarlanan Sistemin JACK Genel Bakış Diyagramı

```
1 public capability ReservedBookReturn extends
2 Capability {
3  #handles external event ReservedBookReturn;
4  #uses plan HandleReservedBookReturn;
5  #agent data ReserveBS ReserveBS();
6 }
```

Listeleme 2: ReservedBookReturn isimli etmen yeteneğine ait JACK kodundan bir parça.

Listeleme 3'te, *HandleReservedBookReturn* planına ait kod parçası bulunmaktadır. Planın içerisinde inanç kümesinde sorgulanan veriler *logical* ve ele alınan olaylar *#handles event* etiketleriyle gösterilmiştir. Plan içerisinde bulunan *relevant()* fonksiyonu, planın çalıştırılması için ele alınan olayın mantıksal kontrollerini, *context()* fonksiyonu ise bu olayın içerik bakımından doğruluğunu kontrol eder. Hem *relevant()* hem de *context()* fonksiyonu "doğru" değer döndürürse planın *body()* fonksiyonu içerisindeki Java kodları çalıştırılacaktır.

```
1 public plan HandleReservedBookReturn extends
2 Plan {
3   logical String $bookName;
4   #handles event ReservedBookReturn ev;
5   #uses data ReserveBS ReserveBS;
6
7   static boolean relevant(ReservedBookReturn
8   ev){…}
9
10  context(){…}
11
12  #reasoning method
13  body(){…}
14 }
```

Listeleme 3: HandleReservedBookReturn isimli plana ait JACK kodundan bir parça.

Listeleme 4'te, *ReservedBookReturn* olayına ait kod parçası yer almaktadır. Bu kod parçasında *ReservedBookReturn* olayının *event* sınıfından türetildiği görülmektedir. Bir olayın gövdesinde *#posted* etiketi varsa, gerekli inanç durumları oluştuğunda olay kendiliğinden çalışır.

```
1 public event ReservedBookReturn extends Event{
2   public String bookName;
3   #posted as
4   set(String b){ bookName = b;}
5 }
```

Listeleme 4: ReservedBookReturn isimli olaya ait JACK kodundan bir parça.

Listeleme 5'te, *ReserveBS* inanç kümesine ait kod parçası verilmiştir. Bu kod parçasında, inanç kümesinde bulunan özellikler *#value field*, inanç kümesindeki birincil anahtar *#key field* ve sorgulamalar *#indexed query* etiketleriyle tanımlanmıştır. JACK'te inanç kümeleri iki sınıftan türetilebilir: *CloseWorld* ve *OpenWorld*. Bir inanç kümesi *CloseWorld* sınıfından türetilmişse, inanç kümesinde yer alan özelliklerin değerleri "doğru" veya "yanlış" olabilir. *OpenWorld* sınıfında ise, "doğru" veya "yanlış" değerlerini alabileceği gibi "*NULL*" değeri de alabilir. Listeleme 5'teki *ReserveBS* inanç kümesi, *OpenWorld*'den türetilmiştir. Öte yandan *ReserveBS* inanç kümesinde iki adet sorgu (*query*) bulunmaktadır. 5, 6 ve 7. satırdaki *get* sorgusu, kitap ismi (*bookName*) ve kullanıcı e-posta adresi (*userEmail*) değişkenlerini parametre olarak alarak bu kitap ismi ve kullanıcı e-posta adresine ait kaydı bulup bu kayda ait kitap dönüş tarihini (*returnDate*) getirir. 8, 9 ve 10. satırdaki *getUserEmail* sorgusu, kitap ismi (*bookName*) ve kitap dönüş tarihi (*returnDate*) değişkenlerini parametre olarak alarak bu kayda ait kullanıcı e-posta adresini (*userEmail*) geri döndürür.

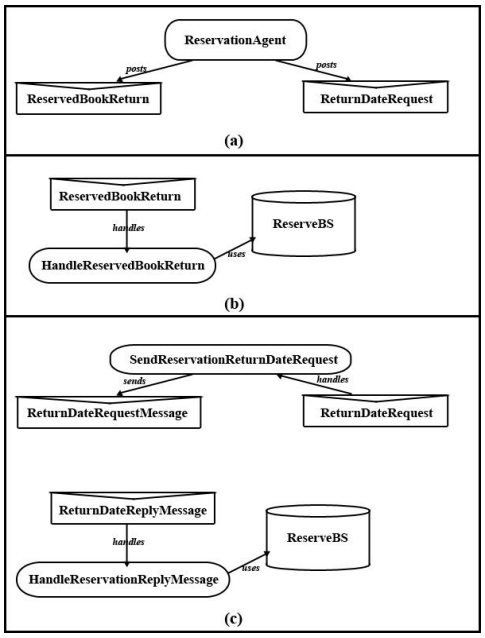
Bildiri kapsamında gerçekleştirilen uygulamanın çalıştırılması sonucu JACK çerçevesinin ara yüzünde elde edilen çıktılar Şekil 10'da verilmektedir.

```
1 public beliefset ReserveBS extends OpenWorld {
2   #value field String userEmail;
3   #key field String bookName;
4   #value field String returnDate;
5   #indexed query get(logical String userEmail,
6   logical String bookName, logical String
7   returnDate);
8   #indexed query getUserEmail(logical String
9   userEmail, String bookName, logical String
10  returnDate);
11 }
```

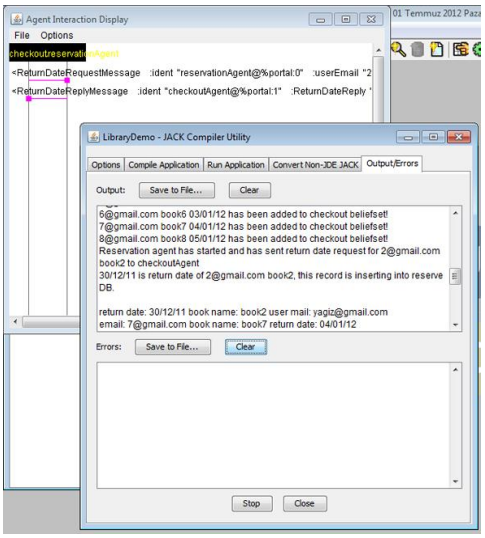
Listeleme 5: ReserveBS isimli inanç kümesine ait JACK kodundan bir parça.

5. Değerlendirme ve Sonuç

Bu çalışmada, örnek bir kütüphane yönetim sisteminin Prometheus metodolojisi takip edilerek oluşturulmuş tasarımı, eksiklikler giderilerek JACK etmen tabanlı yazılım geliştirme ortamına aktarılmıştır. Çalışmada, öncelikle Prometheus metodolojisi takip edilerek örnek senaryo için nasıl bir tasarım oluşturulduğu aşamalarıyla anlatılmış, ardından JACK etmen tabanlı yazılım ortamı kısaca tanıtılmış ve örnek senaryonun JACK ile nasıl gerçeklendiği açıklanmıştır. Bu çalışma sayesinde, Prometheus ve JACK kullanılarak baştan sona kadar bir etmen tabanlı yazılım projesinin nasıl hayata geçirileceği gösterilmiştir. Örnek senaryo her ne kadar küçük olsa da, çok etmenli bir sistemin nasıl oluşturulabileceğine dair fikir vermektedir. Öte yandan, bu çalışma kapsamında üzerinde çalışılan örnek genişletilerek daha kapsamlı bir kütüphane yönetim sistemi geliştirmek de gelecek çalışmalar arasında yer alabilir.



Şekil 9: (a) JACK'deki *ReservationAgent* etmen tasarımı
(b) *ReservedBookReturn* yeteneği
(c) *RequestReturnDate* yeteneği.



Şekil 10: Alınan Sonuçları Temsil Eden Bir JACK Ekran Çıktısı.

Örnek senaryonun gerçeklenmesi esnasında hem Prometheus hem de JACK ile ilgili faydalı olabileceğini düşündüğümüz deneyimler elde edilmiştir. Prometheus ile ilgili deneyimler şu şekilde özetlenebilir:

Çalışmanın hazırlanması esnasında yararlanılan Prometheus metodolojisinin basit ve anlaşılır olduğu görülmüştür. Ancak örnek kütüphane yönetim sisteminin tasarımı esnasında, metodolojideki aşamalar arasındaki geçişlerin net olarak belirtilmediği gözlemlenmiştir. Örneğin mimari tasarım aşamasında, belirlenen etmen tiplerinin oluşturulabilmesi için sistem tanımlama aşamasında elde edilen verilerin ("artifact") yetersiz olduğu düşünülmektedir. Ayrıca, Prometheus'un sağlamış olduğu tasarım aracı olan PDT'nin, kullanımı kolay, basit ve sorunsuz bir araç olduğu; fakat PDT ile oluşturulan tasarımın JACK kodlarına dönüştürülmesi esnasında, PDT'nin kod üretme işlevinin verimli bir kod üretimi gerçekleştirmediği görülmüştür. Örnek olarak PDT'nin JACK proje dosyasını üretmediği ve Prometheus metodolojisinde belirlenen hedeflerin JACK çerçevesinde bulunan *BDIGoalEvent*'ler ile doğrudan eşleşmediği gözlemlenmiştir. Öte yandan PDT'de oluşturulan diyagramların JACK'e doğrudan aktarılamayışı da tasarım sürecini yavaşlatan bir unsurdur. Senaryonun gerçeklenmesi esnasında JACK ile ilgili elde edilen deneyimler ise şu şekilde özetlenebilir:

Uygulama aşamasında Prometheus ve JACK arasındaki bazı kavramsal farklar göze çarpmaktadır. Bu farklar; JACK'te hedeflerin açık olarak tanımlı olmayışı ve Prometheus'ta bulunan algı, mesaj, eylem, protokol gibi kavramların JACK'teki karşılıklarının farklı oluşudur. Bu farklılıkların giderilebilmesi için sistem tasarımının JACK ortamında metodolojiye sadık kalınarak tekrar oluşturulması gerekebilmektedir. Ancak JACK'in sunmuş olduğu tasarım paleti, uygulama geliştirme sürecini hızlandırmakta, iskelet kodların oluşturulması ve uygulamanın anlaşılmasını kolaylaştırmaktadır. Öte yandan JACK yazılım geliştirme platformunun arayüzünün akıllı algı ("*intellisense*") özelliği bulundurmaması, uygulama geliştirmeyi yavaşlatabilecek bir unsur olarak karşımıza çıkmaktadır. Son olarak JACK'in açık kaynak kodlu bir yazılım geliştirme ortamı olmayışı,

çerçeveyi oluşturan kaynak kodlara müdahale etmeyi zorlaştırmaktadır.

Sonuç olarak Prometheus, basit ve kullanışlı bir metodoloji olmasına rağmen, geliştirilmesi gereken metodolojik eksiklikleri olduğunu düşünmekteyiz. JACK'in ise, temelde güçlü bir etmen geliştirme ortamı olarak kullanılabileceğini; fakat bütüncül bir etmen tabanlı yazılım geliştirme ortamı olabilmesi için, akıllı algı desteğinin eklenmesi, sistem çalışırken oluşan bazı çalışma zamanı hatalarının giderilmesi ve görsel arayüzün geliştirilmesi gerektiği söylenebilir.

6. Kaynakça

[1] Wooldridge, M., *An Introduction to Multiagent Systems*, Wiley, 2002.

[2] Tveit, A., "A Survey of Agent-Oriented Software Engineering", *Proc. of the First NTNU CSGS Conference*, May, 2001.

[3] Padgham, L. ve Winikoff, M., *Devoloping Intelligent Agent Systems*, Wiley, 2004.

[4] Bratman, M. E., *Intentions, plans and practical reason*, Harvard University Press, 1987.

[5] Padgham, L. ve Winikoff, M., "Prometheus: A methodology for developing intelligent agents", *Lecture Notes in Computer Science*, 2585:174-185, 2003.

[6] Bellifemine, F., Poggi, A., ve Rimassa, G., "Developing Multi-Agent Systems with a FIPA-compliant Agent Framework", *Software: Practice and Experience*, 31(2): 103-128, 2001.

[7] Pokahr, A., Braubach, L., ve Lamersdorf, W., "Jadex: A BDI Reasoning Engine", *Multi-Agent Programming, Springer*, 15(2):149-174, 2005.

[8] Pokahr, A., Braubach, L., Walczak, A., ve Lamersdorf, W., "Jadex - Engineering Goal-Oriented Agents", *Developing Multi-Agent Systems with JADE*, Wiley, 254-258, 2007.

[9] Howden, N., Ronnquista, R., Hodgson, A., ve Lucas, A., "Jack intelligent agents: Summary of an agent infrastructure", *In proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS at the Fifth International Conference on Autonomous Agents*, Montreal, Canada, 2001.

[10] Winikoff, M., "JACK intelligent agents: An Industrial Strength platform", *Multi-Agent Programming, Springer, Vol. 15:175-193, 2005*.

[11] Iglesias, C., Garijo, M., ve Gonzalez, J., "A Survey of Agent-Oriented Methodologies", *Intelligent Agents V. Agents Theories, Architectures, and Languages, Lecture Notes in Computer Science, Springer, Vol. 1555:185-198, 1998*.

[12] Wood, M. ve Deloach, S. A., "An Overview of the Multiagent Systems Engineering Methodology", *The First International Workshop on Agent-Oriented software Engineering*, Springer, 207-222, 2000.

[13] Wooldridge, M., Jennings, N. R., ve Kinny, D., "The Gaia methodology for agent-oriented analysis and design", Journal of Autonomous Agents and Multi-Agent Systems, 3(3):285-312, 2000.

[14] Mylopoulos, J., Castro, J., ve Kolp, M., "Tropos: Toward agent-oriented information systems engineering", *In Second International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS2000)*, June, 2000.

[15] Jain, P. ve Dahiya, D., "Architecture of a Library Management System Using Gaia Extended for Multi Agent Systems", *Information Intelligence, Systems, Technology and Management, Communications in Computer and Information Science (CCIS)*, 141(7):340-349, 2011.

[16] Rahman, A. ve Mahmood, A. K., "Agent-based simulation using prometheus methodology in evacuation planning", *In Proceedings of International Symposium on Information Technology (ITSim), Vol. 3:1-8, 2008*.

[17] Lucas, A., Corke, P., Rönnquist, R., Sikka, P., Ljungberg, M., ve Howden, N., "Teamed UAVs – A new approach with intelligent agents", *In AIAA Unmanned Unlimited*, 2003.

[18] Lucas, A., Rönnquist, R., Howden, N., Gaertner, P., ve Haub, J., "Intelligent battlespace awareness and information dissemination through the application of BDI intelligent agent technologies", *In SPIE - The International Society for Optical Engineering: Digitization of the Battlespace V and Battlefield Biomedical Technologies II*, April, 2000.

[19] Marc, F., Fallah-Seghrouchni, A. E., ve Degirmenciyan-Cartault, I., "Coordination of complex systems based on multi-agent planning: Application to the aircraft simulation domain", *Second International Workshop on Programming Multi-Agent Systems: Languages and Tools (ProMAS 2004)*, 115–128, July, 2004.

[20] Evertsz, R., Fletcher, M., Jones, R., Jarvis, J., Brusey, J., ve Dance, S., "Implementing industrial multi-agent systems using JACK", *First International Workshop, PROMAS 2003*, Melbourne, Australia, July 15, 2003, Selected Revised and Invited Papers, 18–48, Springer LNAI 3067, 2004.

[21] Mathieson, I., Dance, S., Padgham, L., Gorman, M., ve Winikoff, M., "An open meteorological alerting system: Issues and solutions", *Proceedings of the 27th Australasian Computer Science Conference*, 351–358, Dunedin, New Zealand, 2004.

[22] Gascueña, J. M. ve Fernández-Caballero, A., "Agent-oriented modeling and development of a person-following mobile robot", *Expert Systems with Applications*, 38(4):4280-4290, 2011.

[23] Tamersoy, M., Afsar, B., Erata, F., ve Kardas, G., "Gaia ile Çok-Etmenli Konferans Yönetim Sistemi Analiz ve Tasarımı", *4. Ulusal Yazılım Mühendisliği Sempozyumu (UYMS 2009)*, 83-90, 2009.

[24] Dagdeviren, O., "Öğrenci Bilgi Sistemi Kayıt Senaryosunun Etmen Tabanlı Tasarımı ve Gerçeklenmesi", *5. Ulusal Yazılım Mühendisliği Sempozyumu (UYMS 2011)*, 293-296, 2011.

[25] Deloach, S. A., Wood, M. F., ve Sparkman, C. H., "Multiagent Systems Engineering", *International Journal of Software Engineering and Knowledge Engineering*, 11(3):231-258, 2001.

[26] Örnek Kütüphane Yönetim Sistemi, http://www.cs.rmit.edu.au/agents/pdt/docs/Tutorial.pdf, son erişim tarihi 27.02.2012.