Dağıtık Çakışma Sezme Yaklaşımı

Taner Dursun[1], Bülent Örencik[2]

[1]TÜBİTAK UEKAE
PK 21 41470 Gebze, Kocaeli, Türkiye
tdursun@uekae.tubitak.gov.tr
[2]İstanbul Teknik Üniversitesi
Maslak, İstanbul, Türkiye
orencik@cs.itu.edu.tr

Özet. Bu bildiride, geliştirmekte olduğumuz politika tabanlı yönetim modeli [1] ve bu model içinde kullandığımız politikalar arası çakışma çözme yaklaşımı tanıtılmaktadır. Bizim modelimizde negatif politika kavramı bulunmadığı için **politika tip çakışmaları** [2], oluşmaz. Daha kritik olan **uygulama alanına** özgü çakışmalar [2] ise bu bildiride anlatılan yöntemle yakalanır.

1 Giriş

Politika tabanlı yönetim, geleneksel yönetim mimarilerinin aksine, sistemlerin politika adı verilen üst seviye amaç ifadeleri ile yönetilmesini sağlar. Politikalar, sistemdeki nesnelerin nasıl davranması gerektiğini tanımlar. Yönetim problemini basitleştirmesine rağmen, politika tabanlı yönetim sistemlerinde, politikalar arasında oluşabilecek çakışmalar çözülmeyi bekleyen zor bir problemdir.

Politika çakışmalarının çözülmesi konusunda tüm gereksinimleri karşılayan standart bir çözüm henüz önerilmemiştir. Bu bildiride, POLICE [1] adlı politika tabanlı yönetim modelimizde kullandığımız dağıtık çakışma sezme yöntemimiz tanıtılmaktadır.

Bildirinin ikinci bölümünde bildiriyi anlamak için gerekli bilgi altyapısı, üçüncü bölümde politika belirleme dili, dördüncü bölümde politika uygulama mimarisi sunulmaktadır. Çakışma sezme ve çözme konuları beşinci bölümde, ilişkili Literatür çalışmaları ise altıncı bölümde yorumlanmaktadır. Son bölüm ise sonuçları sunar.

2 Politika Çakışmaları

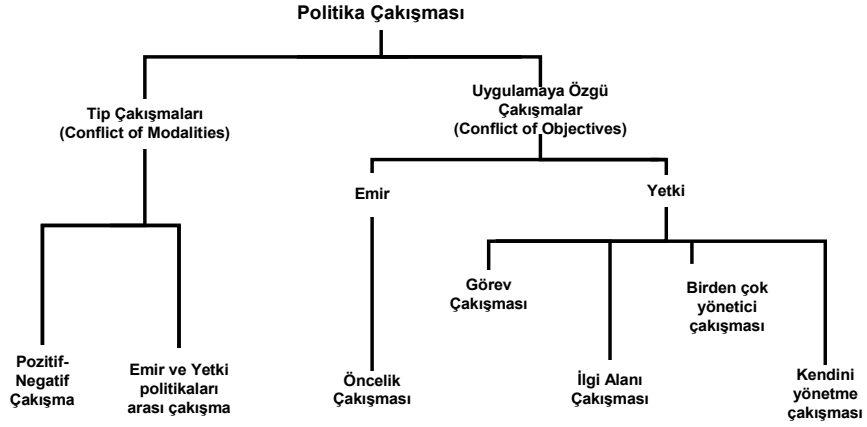
Çakışma, bir politikanın diğerinin eylemlerini engellemesi veya bir başka politika ile etkileşerek yönetilen sistemi istenmeyen durumlara sürüklemesi olarak tanımlanır [2]. Şekil 1'de, Literatürde kabul görmüş genel bir çakışma sınıflandırması [3] gösterilmiştir.
Tip Çakışmaları (Conflict of Modalities)[3], yönetilen sistemdeki aynı varlıklar hakkında hem pozitif hem de negatif **yetki** veya **emir** politikaları tanımlanmış ise ortaya çıkabilir. Bu tür çakışmaların varlığı, politikaların statik analizi ile anlaşılabilir.

Bu tür çakışmalara örnek olarak:

1. bir nesneye başka bir hedef nesne üzerinde bir eylemi hem yapması hem de yapmaması emredilmiş ise,
2. Bir nesneye başka bir nesne üzerinde bir eylemi yapması emredilmiş fakat bu eylemi yapma yetkisi verilmemiş ise

oluşan çakışmalar verilebilir.

Literatürde çakışma çözme konusundaki çoğu çalışma, tip çakışmalarını konu alır. Bu tür çakışmalar, politikalarda yer alan nesnelerin gerçek kimlikleri ve yönetilen sistemin ne olduğundan bağımsız olarak oluşur ve yine aynı şekilde sezilebilirler.



Şekil 1. Politika çakışmalarının sınıflandırılması

Uygulamaya Özgü (Conflict of Objectives) Çakışmalar, politikaların semantiğinden ziyade anlamından dolayı oluşur. Politikanın içinde, sistemin hangi nesne ve eylemlerinin olduğu, sınırlı sistem kaynakları, diğer tüm politikalar gibi dış kriterler, bu tip çakışmaların oluşmasında sebep olabilir. Bu tür çakışmalar, çakışmanın ne olduğunu tanımlayan ek bir bilgi olmaksızın yalnız politikaların tanımı incelenerek yakalanamazlar. Literatürde, çakışmanın ne olduğunu tanımlayan bu ek bilgiyi **meta-politika** [2], [3] kavramı ile modelleyen çalışmalar vardır. **Meta-politika**, tanımlanabilir politikalar üzerinde kısıtlamalar tanımlayan politikadır (politika hakkında politika).

3 POLICE Politika Tanımlama Dili

Politika tabanlı yönetimi kolaylaştırmak için kolay anlaşılır bir politika tanımlama dilinin yanı sıra, politikaları dağıtacak, uygulatacak ve koordinasyonunu sağlayacak modüller de gereklidir. POLICE modeli, genel amaçlı yönetim politikaları tanımlamaya olanak sağlayan nesneye dayalı bir dil içerir. Modelimizde iki temel politika tipi vardır:

- **yetki** (*authorization):* sistemdeki bir etmene, diğer nesneler üzerinde bir eylemi gerçekleştirme yetkisi veren politika,
- **emir** (*obligation):* sistemdeki bir etmenin bir eylemi gerçekleştirmesi zorunluluğunu tanımlayan politikalardır. Bizim modelimizde emir politikasının, bu eylemi gerçekleştirebilmesi için etmene gerekli izni de (bir yetki politikası tanımlanmasına gerek kalmadan) otomatik olarak verdiği kabul edilir.

Literatürde bu iki temel sınıf, pozitif ve negatif olmak üzere daha alt gruplara da bölünür. [3], [4]. Ancak biz bu alt sınıfların, politikaların uygulanmasını zorlaştırdığını düşündüğümüz için modelimizde negatif politikaları içermeyerek politika çakışmalarını daha kolay kotarabilme olanağını elde ettik. Çünkü **tip çakışmalarının** temel nedeni olan pozitif-negatif politikaların tutarlılığını sağlama problemi kendiliğinden elenmiştir.

Negatif politikalar her ne kadar, politika uygulanmasını karmaşıklaştırsa da bazı sistemlerde kullanılmaları mantıklı olabilir. Örnek olarak, negatif yetki politikalarının, etmenlerin çeşitli erişim yetkilerini geçici olarak kaldırmak için kullanılması verilebilir. POLICE modeli, bu tür bir yetki kısıtlamasına olanak sağlamak için negatif politikalar yerine, izin verilmiş bazı eylemleri sadece bazı etmenler bazında sınırlamaya yarayan NEVER ve ALWAYS anahtar kelimelerini kullanır. Modelimizde her eylem, bir politika ile aksi belirtilmedikçe yasaklıdır ("Permission-Based Regime").

Modelimizde, yönetilen sistemin modülleri birer nesne ve bu modüller ile etkileşim ise bu nesnelerin metodlarını çağırma olarak düşünülmüştür. Sistem içindeki insan kullanıcılar, yazılım etmenleri gibi aktif varlıklar, **aktör** (**actor)** olarak adlandırılır. Aktörler diğer nesneleri (hedef nesneler**),** metodlarını çağırarak kullanırlar.

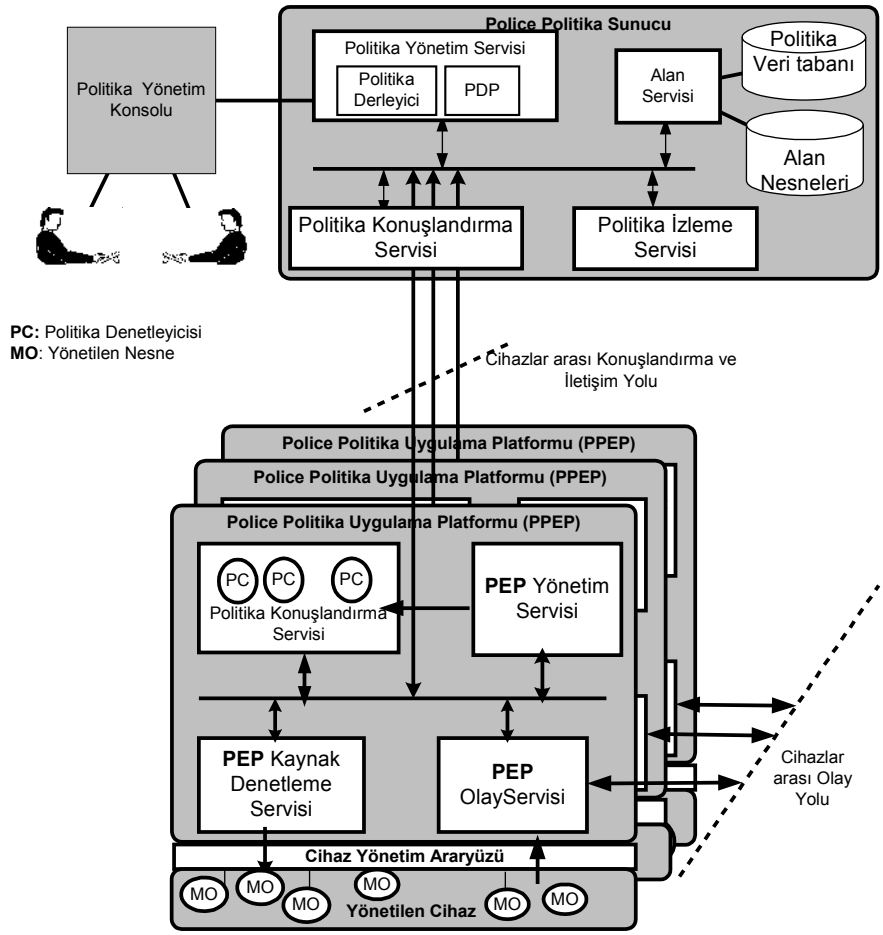
4 POLICE Politika Konuşlandırma Altyapısı Elemanları

Politika uygulama altyapısı Şekil.2'de gösterilmiştir. **Politika Sunucusu**, poltika yönetiminin merkezidir. Bu sunucu içinde bulunan **Politika Yönetim Servisi,** politikalar için konuşlandırma (deployment) nesneleri oluşturur. **Politika**

Konuşlandırma Servisi ise bu nesnelerin **gezgin etmenler** şeklinde (mobile agent [5]) politika uygulama noktalarına (**PEP** :Policy Enforcement Point) dağıtılmasını koordine eder. Gezgin etmenler, politika nesnelerini, kendilerine verilen listedeki uygulama noktalarına ulaştırmaktan sorumlu olan yazılım varlıklardır.

Modelimizde hem yetki hem de emir tipindeki politikalar, **Politika Denetleyicisi** (**PD**) adı verilen etmenlerin kontrolünde uygulanır. Gezgin etmenler tarafından uygulama noktalarına ulaştırılan konuşlandırma nesneleri, ilgili **PD'**ye eklenir. PD'ler bu nesneleri yapılarına ekleyerek davranışlarını şekillendirirler. Sistemdeki aktörlerin eylem girişimleri, o aktörlerden sorumlu olan **PD**'nin kontrolünden geçmek zorundadır.

Hem **emir** hem de **yetki** tipindeki politikalar, aynı yolla, aktörlerin yakınındaki **PD'**lere dağıtılır. Aktörlere göre tek tip dağıtım, modelimizin mimarisini daha basit yapar. Yetki politikaları, politikadan etkilenen hedef nesneler yerine, aktör nesnelerin yakınında uygulanır. Böylece izin verilmemiş eylemeler, kaynağında (aktörlerin yanında) yakalanarak sistem kaynaklarının boş yere harcanması önlenmiş olur. Bu ayrıntı, modelimizi, bu alanda çok iyi bilinen Ponder modelinden farklı yapar [6, 7, 8]. POLICE felsefesi, her aktörün başına bir **polis memuru** (Politika Denetleyisi) atamak suretiyle hedef nesneleri korumaya dayanır. Modelimiz ismini buradan almaktadır.



Şekil 2. Police altyapı mimarisi

Aktörlerin, diğer sistem nesnelerine erişim istekleri (metod çağırımlarını), politika uygulama noktalarında (PEP) bulunan **Kaynak Denetleyici Süreci'**nin (KDS) üzerinden geçer. KDS, bu girişimleri, aktörün sorumlusu olan **PD**'ye yönlendirerek erişim izninin (yetki politikalarının) kontrol edilmesini sağlar. PD'ler, bir aktöre yaptırmak istedikleri (emir politikası ile tanımlanmış) eylemleri tetiklemeden önce politikada belirtilen koşulların oluştuğundan emin olmalıdırlar. PD'ler koşulların oluşup oluşmadığının takibini Olay Servisi adlı sürece yaptırırlar. Olay Servisi'nden, koşulların oluştuğunu belirten işareti alan PD ilgili aktörleri eyleme zorlar.

Olay Servisi, yönetilen düğümlerde oluşan olayları izleyerek bu olayları dinlemek için kendisine kayıt yaptırmış olan PD'lere iletir. Dağıtılmış sistemdeki bütün düğümler üzerindeki olay servisleri, bir iletişim yolu üzerinden haberleşen tümleşik bir servis oluştururlar. POLICE modelinin kendine has diğer bir özelliği ise hem emir hem de yetki politikalarının tek tip (geçerliliği, olaylar şekline dönüştürülerek takip edilen alt bölümlerden oluşan) koşul bölümü içermesidir.

Alan Servisi: Alan (Domain) kavramı, geniş bir sistemde nesneleri çeşitli özelliklerine göre gruplama esnekliği sağlar. Police politikaları bir nesne grubuna veya tek tek nesnelere uygulanabilir. Alan Servisi, sistemdeki nesnelerin tanımlarını ve tanımlanmış politikaları tutar. Politikalar uygulanırken yönetilen nesnelerin hem çeşitli özelliklerinin (attribute) değerleri sorgulanır hem de çeşitli metodları çağrılır. Bu nedenle Politika Yönetim Sunucusu, politikalar tanımlanmadan önce, yönetilen sistemdeki nesnelerin detaylı bilgisine sahip olmalıdır. Bu amaca, **Alan Servis**i'nde tutulan ve sisteme yeni nesneler eklendikçe güncellenen **model sınıflar** veritabanı ile ulaşılır. Model Sınıflar, yönetilen nesnelerin Alan Servisi'ndeki temsilcileridir. Her model sınıfı, temsilcisi olduğu yönetilen nesnenin arayüzlerinin (metodlarını) tanımını içerir. Alan Servisi'nde, yönetilen sisteme ait olarak tutulan bilgilerin tamamına **Sistem Bilgi Modeli (SBM)** adı verilir. SBM, model sınıflarını ve aralarındaki ilişkileri içerir.

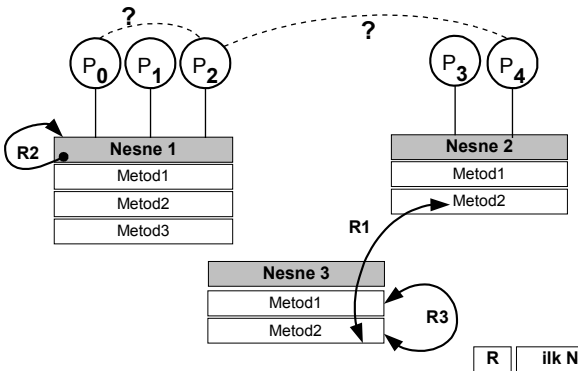
5 POLICE Çakışma Kotarma Yöntemi

[3],[4],[9] nolu çalışmalarda, ortak hiç bir nesne (aktör, hedef) içermeyen iki politika (ayrık politikalar) arasında çakışma ihtimalinin olmayacağı iddia edilmektedir. Ama bu iddia uygulamaya özgü çakışmalar için geçerli değildir. Örneğin tamamen farklı nesneler hakkında tanımlanmış iki politikanın görünüşte hiç bir şekilde çakışmayacağı düşünülebilir. Fakat bu politikaların tanımında geçen nesneler arasında uygulamaya özel bazı ilişkiler bulunabilir (kapalı çakışma ihtimali). Bu durumda politikaların uygulanması uygulamaya özel tipteki çakışmalara neden olabilir. Bu tür çakışmaları yakalamak için, nesnelerin isimleri yeterli değildir; yönetilen sistemin yapısı (nesneler ve ilişkileri) hakkında bilgi gereklidir.

Meta-Politika temelli yaklaşım, uyulamaya özgü çakışma gereksinimlerini tam olarak karşılamaz. Zira, sistem yöneticilerinin, yürütme anında çakışmaya sebep olabilecek tüm olasılıklar hakkında meta-politikalar tanımlaması gerekir. Bazı ihtimalleri gözden kaçırmaları olasıdır. Dahası, çoğu durumda politikaların birbiri ile ilişkili olması, yürütme anında aralarında mutlaka çakışma olacağı anlamına gelmez.

Negatif politika kavramı bulunmadığı için modelimizde politika tip çakışmaları sorunu yoktur. Uygulamaya özgü çakışmalar konusunda ise meta-politika yöntemini kullanan çalışma grupların aksine, modelimizde, yönetilen sistemin nesneleri arasındaki ilişkilere dayanan yeni bir *dağıtık çakışma kotarma yöntemi* geliştirilmiştir.
Modelimizin çalışması için yönetilen sistemdeki nesneler ve aralarındaki tüm ilişkilerin, politikalar çalıştırılmadan önce **Alan Servisi** içerisinde tanımlanmış olması gereklidir.. Nesneler arasındaki ilişkiler, kendilerine uygulanan politikalar arasındaki ilişkileri gösterir. Nesneler arasında tanımlı bu ilişkileri incelemek suretiyle, politikalar arasındaki doğrudan veya dolaylı ilişkiler belirlenebilir. **PD'**ler, yürütme anında politikalar arasındaki ilişkileri inceleyerek muhtemel çakışmaları yakalarlar. Nesneler arasındaki her ilişki tipine karşılık, PD'lerin sistemde çakışma ihtimalini araştırmalarını sağlayan kod parçacıkları da hazırlanır. Aşağıda verilen detaylı örneklerde, Literatürde [2], [4], [8] tanımlanmış, uygulamaya özgü çakışma tiplerinin bizim modelimizde nasıl kotarıldığı açıklanmıştır. Aşağıdaki politikaları içeren bir senaryo varsayalım ($\mathbf{P_i}$ politikası, $\mathbf{N_i}$ aktör_nesnesinin → $\mathbf{N_j}$ hedef_nesnesinin $\mathbf{m_i}$ eylemine erişimini düzenler) :

- $P_0 : N_1 \rightarrow N_2.m2$
- $P_1 : N_1 \rightarrow N_2.m1$
- $P_2 : N_1 \rightarrow N_3.m2$
- $P_3 : N_2 \rightarrow N_3.m2$
- $P_4 : N_2 \rightarrow N_3.m1$



İlişki Tablosu

R	ilk Nesne	ikinci Nesne	ilişki
R1	N3.metod2	N2.metod2	mutex
R2	N1	N2, N3	onlyOne
R3	N3.metod1	N3.metod2	related

Şekil 3. Nesne ve Politika ilişki grafiği

- **Görev Çakışması:** Mutlaka, farklı iki aktör tarafından gerçekleştirilmesi gereken bir eylemin aynı aktör tarafından yürütülmesi. N_2**.m2** metodu bir çeki imzalama işlemi, N_3**.m2** ise çek ödemelerine onay verme işlemi olsun. Çakışmayı önlemek için sistemde, bu iki metodun farklı aktörler tarafından çağrıldığı garanti edilmelidir. Bu amaçla Şekil.3'de **R1** olarak gösterilen bir ilişkisi tanımlanmıştır. Bu durumda, aynı aktöre (N_1) bu iki metodu yürütme emri veren P_0 ve P_2 politikaları çakışmaya neden olacaktır.
- **Öncelik Çakışması:** Sistemdeki kaynak (hedef nesneler) miktarının sınırlı olduğu durumlarda oluşur. Modelimizde nesne erişimleri Kaynak Denetleyici Sürecin kontrolünde olduğu için bu tür çakışmalar elenir.
- **Birden Çok Yönetici Çakışması:** Aynı nesneyi kullanan aktörlerin eriştikler metodlar birbiri ile bağımlı ise oluşabilir. N_3**.m1** ve N_3**.m2** birbiri ile bağımlı olsunlar (mesela her iki metod ortak bir değişkenin değerini değiştirin). Dolayısıyla bu iki metoda ulaşan farklı aktörlerin faaliyetleri birbirini etkileyebilir. Bu durum **R3** ilişkisi ile gösterilmiştir. Bu denele P_2 ve P_4 politikaları dikkatlice yürütülmelidir.
- **Kendini Yönetme Çakışması:** Bir aktöre kendisi ile ilgili politika tanımlama veya tanımlı olanları değiştirme yetkisi veren politika bulunması ile oluşur. Politikalaın uygulanması Politika Denetleyicilerinin yönetiminde yürütülür. Dolayısıyla aktörlerin emredilen eylemi gerçekleştirmeme veya politikaları değiştirme şansları olmaz.
- **İlgi Alanı Çakışması:** Bir aktörün iki farklı nesne üzerindeki metodları çağırması gerektiğinde eğer bu iki nesne üzerindeki işleri tutarlı olarak yapması imkansız ise oluşur. N_1 'in {N_2, N_3} kümesinden sadece birisi ile ilgili işleri yürütmesi zorunlu olsun. Bu durm **R2** ile gösterilmiştir. Yani N_1 yalnızca birisinin metodunu çağırabilir.

6 İlişkili Çalışmalar

Ponder modeli, tip çakışmalarını yakalamak üzere politikaların statik analizi için bir çakışma analiz aracı içerir [2], [9]. Bu araç çakışma çözme için politikalar arası öncelik ilişkileri tanımlar. Tanımlanan her yeni politika ile birlikte tekrarlanan merkezi statik analiz, politika sayısı arttıkça zaman açısından maliyetli olur. Uygulamaya özel çakışmalar için ise meta-politika kavramını kullanır.

Diğer yandan, [10], [11], [12], [13]'deki gibi, politikaların mantık-tabanlı tanımlanması için de önemli çalışmalar vardır. Amaç, politikaların mantık formatında gösterilimi gerçekleştirilince var olan formal hesaplama yöntemleri, teoremler, araçlar, politikaların tutarlılığının analizinde kullanabilmektir [10], [14]. Ancak bu çalışmalar sistem gerçekleme çalışmalarına kolayca dönüştürülememektedir.

Politikaları IETF formatı [15] kullanarak tanımlamak, mantık tabanlı araçları, uzman sistemleri kullanmaya olanak sağlayabilir. Ancak, bu tür formal metodları kullanarak politika analizi, kompleks sistemlerde kabul edilemez sürelerde gerçekleşebilir. Dahası politikalar tanımlanırken statik analiz yürütülmesi tüm çakışmaları yakalamayabilir. Çünkü çakışma olarak değerlendirilen bazı durumlar, yürütme anındaki koşullara göre belki de hiç gerçekleşmeyecektir.

Diğer bir çalışma [17], çoklu-etmenli (multi-agent) mimarilerde çakışmaları konu almaktadır. Bu tür sistemlerde etmenler, paylaşılan görevleri işbirliği içerisinde yerine getirir. Paylaşılan görevler arasındaki muhtemel bağımlılıklar, etmenler görevleri yürütürken çakışmalara sebep olabilir ve bu çakışmalar etmenler için tanımlanan politikalarla çözülür. Çakışma çözme sürecinde etmenler arasında devam eden bir pazarlık sonucu bazı görevlerin sırası ve zamanı değiştirilir. [17]'da çakışan işleri çözmede kullanılan yapının, politika tabanlı yönetim sistemlerinde çakışan politikaları çözmek için de kullanılabileceğini düşünüyoruz. Dahası, bizim **SBM** kavramımızın çoklu-etmen mimarisinde ihtiyaç duyulan altyapı için kullanılabileceğini değerlendirmekteyiz. Görevler arasındaki ilişkiler, yönetilen nesneler arasındaki ilişkiler olarak modellenebilir.

7 Sonuçlar ve Planlanan Çalışmalar

Politika tanımlama ve gerçekleme konusunda iki ana seçenek vardır; Politikaları programcıklar olarak veya nesneler olarak gerçeklemek [17]. Politikaları kod parçacığı olarak gerçeklemek daha esnektir. Ama, bu durumda iki politikanın (iki kodun) birbiri ile tutarlı olduğunun analiz oldukça zordur. Politikaların nesneler olarak ifade edilip uygulama etmenlerince yorumlanması yaklaşımı, politika analizini basitleştirir [9], [17].

Bazı yazarlar [2], [18] politika tanımını basit tutmayı tercih ederken genel eğilim, analizler zor bile olsa kod şelindeki politikaların kullanılmasının var olan politika sistemlerindeki sınırlamaları aşmak için şart olduğu yönündedir.

Politika tutarlılığını analiz alanında hem nesne hem de kod şeklindeki politikalar için tüm ihtiyaçları karşılayan çözümler henüz geliştirilmemiştir. Acak bu çalışmada tanıtılan yöntemin bu eksikliği gidereceğini düşünmekteyiz. Zira

yöntemimiz, politikanın karmaşıklığından bağımsz bir çakışma çözme sağlamaktadır. Bununla birlikte Police modelinin aşağıdaki alanlarda daha da geliştirilmesi gerekmektedir:

- Akıllı etmenler kullanarak politikaları uygulama ve nesneler arasındaki ilişkileri inceleyerek çakışmaları sezmek,
- Modelin çeşitli uygulama alanlarında denenmesi,
- Politikalar ve nesneler arasındaki ilişkileri ve aralarındaki çakışmaları daha genel bir modele oturtmak.

Kaynakça

1. T.Dursun, B.Örencik, "POLICE: A Novel Policy Framework", 18.Uluslararası Bilgisayar ve Enformatik Sempozyumu, Springer-Verlag, Antalya, Türkiye, Kasım 2003
2. E. Lupu, M. Sloman: Conflicts in Politika-based Distributed Systems Management, IEEE Trans. On Software Engineering, Vol 25. No 6, 1999, s. 852-869
3. Moffett J.D: Politika Hierarchies for Distributed Systems Management, IEEE JSAC Ağ Yönetimi Özel Sayısı, Vol 11, No. 9, 1993
4. M. Sloman: Policy Driven Management for Distributed Systems, Journal of Network and Systems Management, Plenum Publising. Vol 2, No.4, 1994
5. V. A. Pham, A. Karmouch: Mobile Software Agents: An overview, IEEE Communication Magazine (1998) 26-37
6. Damianou N., N., Dulay, E., Lupu, M., Sloman: Ponder: The Language Specification – V2.2. Research Report DoC 2000/1, Imperial College of Sci. Tech. and Medicine, London, 2000
7. N.Dulay, E.Lupu, M.Sloman, N.Damianou: A Policy Deployment Model for the Ponder Language, IEEE/IFIP Int..Symp. on Integrated Network Management, 2001
8. Sloman, E. Lupu: Security and Management Policy Specification, IEEE Network Policy based management special issue, 2002, s. 10-19
9. Moffett J.D., Sloman M.S: Politika Conflict Analysis, Jrnl. of Org. Computing, 1993
10. L. Chalvy, F. Cuppens: Analyzing, Consistency of Security Policies, IEEE Symp. Security and Privacy, 1997
11. Ortalo, R.: A Flexible Method for Information System Security Politika Specification. 5th European Symposium on Research in Computer Security (ESORICS 98), Springer-Verlag, Louvain-la-Neuve, Belçika, 1998
12. Barker, S.: Security Politika Specification in Logic. International Conference on Artificial Intelligence, Las Vegas, ABD, 2000
13. Ahn, G.-J., R. Sandhu: The RSL99 Language for Role-based Separation of Duty Constraints, Fourth ACM,Workshop on Role-Based Access Control, Virginia, ACM Press, 1999
14. J. B. Michael: A Formal Process for Testing the Consistency of Composed Security Politikaları, Dept. of Info. and Software Sys. Eng., George Mason Univ., Fairfax, VA, 1993
15. R. Yavatkar, D. Pendarakis, R. Guerin: A Framework for Policy-based Admission Control. RFC 2753, Intel, IBM, U. of Pensylvania, 2000
16. T. Wagner, J. Shapiro: Multi-Level Conflict in Multi-Agent Systems, Proc. of AAAI Workshop on Negotiation in Multi-Agent Systems, 1999
17. P. Martinez.: Using the Script MIB for Policy-based Configuration Management, IEEE NOMS, 2002, s. 203-219
18. D.C. Verma, S. Calo, K. Amiri: Policy-Based Management of Content Distribution Networks, IEEE Network Politika tabanlı yönetim özel sayısı, 2002, s. 34-39