Yazılımlar arası Kural Dışı Durumları İşleme için Etmen Tabanlı bir Yaklaşım *

Albert ÖZKOHEN [1] **Pınar YOLUM** [2]

[1,2] Boğaziçi Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bebek-İstanbul

[1]e-posta: aozkohen@elitsoft.com.tr [2]e-posta: pinar.yolum@boun.edu.tr

Özet

Bu makale yazılım mühendisliğinde oluşabilecek kural dışı durumları katılımcıların oluşturdukları taahhütlerin ihlalleri olarak modellemeyi önermektedir. Katılımcılar etkileşimlerinden sonuç çıkarabilen ve diğer etmenlerle haberleşebilen akıllı etmenler olarak modellenmiştir. Etmenler arası haberleşmeler taahhütler olarak gösterilmektedir. VE/VEYA ağaçları kullanılarak çok katılımcılı taahhütler daha küçük, az katılımcılı taahhütlere bölünebilir. Buna ek olarak taahhütlerde bahsedilen durumlar ve durumlar arasındaki ilişkiler bir OWL ontolojisiyle gösterilmiştir. VE/VEYA ağaçlarını ve OWL ontolojisini kullanarak etmenler kural dışı durumların çaresine bakabilir ve bazı durumlarda kural dışı durumları, daha olmadan, fark edebilirler. Önerilen yöntem, yazılım mühendisliğine sıkça konu olan tedarik zincirlerine uygulanmıştır.

Abstract

This paper proposes to model exceptions in software engineering as violations of commitments of participants. The participants are represented as intelligent agents that can reason about their interactions and communicate with other agents. The communications of the agents lead to creation and manipulation of commitments. Commitments that involve several participants are broken into smaller commitments using AND/OR trees. Further, the commitment conditions and their relations are captured in an OWL ontology. By reasoning based on the AND/OR tree and a given ontology, agents can handle exceptions when they occur and in some cases even detect exceptions before they take place.

1. Giriş

Yazılım mühendisliği etkin, verimli ve daha önemlisi emniyetli yazılımlar tasarlamayı ve geliştirmeyi hedefler. Oysa birçok pratik çalışma, geliştirilen yazılımların hemen hemen hiçbir zaman istendiği şekilde hareket etmediğini, değişik zamanlarda kural dışı durumların ortaya çıktığını göstermektedir. Geliştirilen yazılımlar, başka yazılımlarla haberleşme içerisinde veya başka yazılımlara işlevsel olarak bağlıysa kural dışı durumların oluşma ihtimali de artar [1]. Genelde, yazılımlara karşılaşabilecekleri kural dışı durumlar ve bu durumların nasıl giderileceği bilgisi verilmişse, sistemler belirtilen çözümleri uygulayarak kural dışı durumdan, kurallara uygun bir duruma geçebilirler. Hâlbuki birçok kural dışı durum önceden kestirilemez. Burada yazılımın alt kademelerinde oluşan kural dışı durumlar (sıfıra bölme gibi) değil, daha üst seviyelerde olan (bir

üretici firma otomasyon yazılımının müşterilere doğru fiyatı göndermemesi gibi) kural dışı durumlar düşünülmektedir. Bu tür kural dışı durumları işlemek için, yazılımların önce bir sorunun varlığını, sonra sorunun nereden kaynaklandığını, sonrasında ise sorunun çözümünü, kendi başlarına (yazılımı geliştiren kişilerin müdahalesi olmadan) bulmaları gerekir. Kural dışı durumları bu şekilde işleyebilmek için yazılımların zeki ve inisiyatifli olmaları şarttır. Bunu sağlayabilmek için bu bildiri, yazılımları algılayan, sonuç çıkaran, buna bağlı olarak hareket edebilen ve diğer yazılımlarla iletişim kurabilen özerk *etmenler* olarak modellemeyi önermektedir [2,3]. Böylece, etmenler sadece sıralı işlemleri icra etmeyecek, aynı zamanda kendi işlemleri hakkında çıkarsamalar yapabileceklerdir. Başka bir deyişle etmenler çalışmalarındaki kural dışı durumları algılayıp bunları çözümlemek için yöntemler geliştireceklerdir.

Örnek uygulama alanı olarak, yazılım mühendisliği problemlerine sıkça konu olan tedarik zincirlerinin otomasyonunu alalım [2]. Tedarik zincirleri müşteriler, tedarikçiler, ulaşım sağlayıcılar, fabrikalar, depolama birimleri gibi bileşenlerden oluşur. Tedarik zincirlerinin otomasyonu ise bu birimlerin fonksiyonlarının bir yazılımla takip edilmesi ve işlevlerinin bir kısmını otomatik ya da yarı otomatik yapmaları anlamına gelir. Bu birimler birbirleriyle sıkça habe[1]rleştiği ve bir çok işlevleri birbirlerine bağımlı olduğu için kural dışı durumların oluşması hayli mümkündür. Yapılan çalışmalarda, eğer doğru şekilde müdahale edilmezse, tedarik zincirinde oluşan kural dışı durumların şirketlerde zamansal ve parasal kayıplara yol açabildiğini göstermiştir [4]. Kural dışı durumlar verimlilik, maliyet artışları, müşteri tatmini ve kalite yönetimi gibi çeşitli boyutlarda karşımıza çıkmaktadır [5]. Tedarik zinciri yönetimini otomasyon altına almak için çeşitli yaklaşımlar vardır. Fakat bu yaklaşımlardan hiçbirisi bugüne kadar tedarik zinciri operasyonlarında oluşabilecek kural dışı durumları otomatik olarak yakalayıp çözemez.

Tedarik zinciri yönetim sistemlerindeki kural dışı durumların otomasyon yoluyla işlenmesi ile ilgili halen süregelen araştırmalar bulunmaktadır [5,6,7]. İstisnaların işlenmesi, tanımlı ve genişleyebilen kural kümeleri altında karmaşık çıkarsama faaliyetleri gerektiren bir süreçtir. Mevcut durumda, kural dışı durumlar çalışan profesyonellerin beyin gücü kullanılarak halledilmektedir. Bu kişiler mevcut tedarik zinciri yönetim araçlarını kullanmaktadırlar. Fakat bu sonuç pahalı ve verimsizidir, zira insana dayalı operasyonlar otomasyon yaklaşımına göre daha zaman alıcı ve hataya açıktırlar.

Bu bildiri şu şekilde devam etmektedir: Bölüm 2 kural dışı durumların taahhütler ve VE/VEYA ağaçları kullanarak modellenmesini anlatır. Bölüm 3 tedarik zinciri uygulamalarında oluşabilecek senaryoları anlatmak için kullanılabilecek bir ontoloji geliştirir ve yazılımların bu ontolojiyi nasıl kullanabileceklerini anlatır. Bölüm 4 bu bildiride geliştirilen metodu diğer araştırmalarla karşılaştırıp, tartışır. Buna ek olarak, Bölüm 5 ifade edilen fikirleri birleştirip, taahhütlerin VE/VEYA ağacı şeklindeki gösterimini ve bir bilgi tabanı oluşturmak amacıyla taahhütler arasındaki çıkarsamaya yardımcı olacak bir ontolojiyi sunar.

2. Kural Dışı Durumların Modellenmesi

Tedarik zinciri sistemlerinde çok çeşitli kural dışı durumlar oluşabilir. Huhns ve öbürleri [5] kural dışı durumları sınıflara ayırmışlardır. Bu gruplar, son günlere uyulmaması, ürün tanımları, ödemeler, sevkıyatlar, depolama seçenekleri, siparişler gibi farklı sınıflarda oluşturulabilir. Bunlar birbirinden tamamen bağımsız ya da birbirini karşılıklı dışarlayan sınıflar değildirdirler. Bir kural dışı durum birden fazla sınıfa ait kural dışı durum olabilir. Örneğin bir siparişte bir kural dışı durum yanlış ürün tanımından dolayı oluşabilir, bu da depolama sorunu oluşturabilir. Tüm bu sınıfları

[1] Bu çalışma Boğaziçi Üniversitesi Araştırma Fonu (BAP05A104) tarafından desteklenmektedir.

tanımlamak için ontoloji adını verdiğimiz bir bilgi altyapısı kullanmaktayız. Ontoloji kural dışı durumları modellemek için önerdiğimiz taahhütlerin arasındaki bağlantıları tanımlamaya yardımcı olacaktır. Bu kural dışı durumları ve sonuçlarını *bulgu-teşhis-reçete* zinciri şeklinde n-ayaklı ağaç türü yapılarla gösteriyoruz. Böylece içerik gösterimi için bir altyapı oluşturabiliriz. Bu gösterime kısaca taksonomi denir [6]. Burada teşhis seviyesi özel bir önem hak etmektedir; zira teşhis içeriğe bağlı olarak değişiklik gösterebilir. Hareketler de içerik-bağımlı olup taksonomimizdeki (ya da ağaç yapımızdaki) seviyeyle doğrudan ilgilidir. İstisna sınıflarında gruplanabilecek kural dışı durumlar önceden tanımlanmış bir hareket kümesini (hareket planı) iç çıkarsama amacıyla tetiklerler.

2.1. Taahhütler

Kural dışı durumları anlayabilmek için önce kurallara uygun durumları göstermek gerekmektedir. Veritabanı araştırmalarından yola çıkarak, etmenlerin aralarındaki iletişimleri *taahhütler* (commitments) olarak gösteriyoruz. Örneğin, bir üreticinin alıcı firmaya ürünü 3 Haziran'a kadar teslim edeceğini taahhüt etmesi yazılımlar arasında beklenen ve istenen bir iletişimi göstermektedir. Bu taahhüdün edildikten sonra yerine getirilmemesi ise kural dışı bir durum yaratacaktır. Bu sebeple, etmenler arası iletişimi taahhütlerle göstermeyi ve taahhütlerin yerine getirilmediği durumları da kural dışı durumlar olarak kabul ediyoruz.

Taahhütler, bir veri yapısı şeklinde düşünülebilir [8]. Bu yapıda *borçlu* taraf, *alacaklı* tarafın lehine bir *koşul*u *zaman* dolmadan yerine getirmekle sorumludur. Bu taahhüt aşağıdaki notasyonla gösterilebilir [9]:

T (borçlu, alacaklı, koşul, zaman)

Taahhüt yaratıldığında *borçlu* taraf, *alacaklı* taraf lehine *koşul*u belirtilen *zaman*'dan önce yerine getirmekle mükelleftir. *Koşul* doğru olunca, *taahhüt* başarıyla sonlandırılır.

Şartlı taahhüt [9] resme bir ön şart ekler. Şartlı taahhütler iki tarafın birbirlerine karşılıklı taahhütte bulunmalarını gerektirir. Bu oluşumda *borçlu* taraf bir *koşulu alacaklı* taraf lehine eğer bir *önkoşul* yerine getirilirse taahhüt etmektedir. Bunu aşağıdaki şekilde gösterebiliriz:

ŞT (borçlu, alacaklı, önkoşul, koşul, önkoşul zamanı, koşul zamanı)

Yani, böyle bir şartlı taahhüt yapan borçlu, alacaklının önkoşul zamanına kadar önkoşulu yerine getirdiği takdirde, koşulu koşul zamanından önce yerine getireceğini taahhüt eder. Taahhütler etmenler (insan ya da yazılım) arası iletişimin her evresinde mevcuttur. Taahhütler, etmenlerin faaliyetleri (dış dünyayla etkileşimleri ya da birbirleriyle mesajlaşmaları) sonucunda yaratılırlar. Örneğin, bir tedarik zinciri üzerinde bir üreticinin mallarını 50 YTL'den sattığını belirtmesi aslında, 50 YTL veren ilk bireye malını vereceğini taahhüt etmesi anlamına gelir. Yaratılan taahhütler zaman içersinde yerine getirilebilir (discharge), gözden geçirilebilir, iptal edilebilir, vb. Bu bildirinin ana konusu olan kural dışı durumları çalışmak için yerine getirilen taahhütlerle, yerine getirilmeyen taahhütleri birbirlerinden ayırabilmek yeterlidir. Taahhütlerin yerlerine getirildiği durumda beklenmedik, kural dışı bir durum gerçekleşmez. Bu sebeple kural dışı durumları fark etmek için taahhütlerin yerlerine getirilmediği durumları tespit etmek gereklidir. Biraz önceki örnekten devam edersek, bir alıcı 50 YTL ödemesine karşın, satıcı taahhüdünü yerine getirmezse (ürünü alıcıya vermezse) beklenmedik, kural dışı bir durum olmuş demektir. Dolayısıyla yukarıda açıklandığı şekliyle taahhütler iş iletişiminde önemli bir role sahiptirler. Bu yüzdendir ki, kural dışı durumların taahhütler (ya da taahhüt ihlalleri) yoluyla modellenmesi iş iletişiminin doğal sonucu olarak önümüze çıkmaktadır.

2.2. VE/VEYA Ağaçları

Birçok zaman yazılımlar arası olan taahhütler daha küçük taahhütlere bölünebilir. Örneğin, bir üreticinin bir alıcıya belirli bir ürünü 15 gün içinde teslim edeceğini taahhüt etmesi aslında (1) üreticinin 15 günden önce ürünü hazır edeceği ve (2) üreticinin hazır ettiği ürünü postaya zamanında vereceği anlamına gelir. Bu şekilde daha basit taahhütlere bölünebilen taahhütleri, VE/VEYA ağaçlarıyla [10] göstermeyi öneriyoruz.

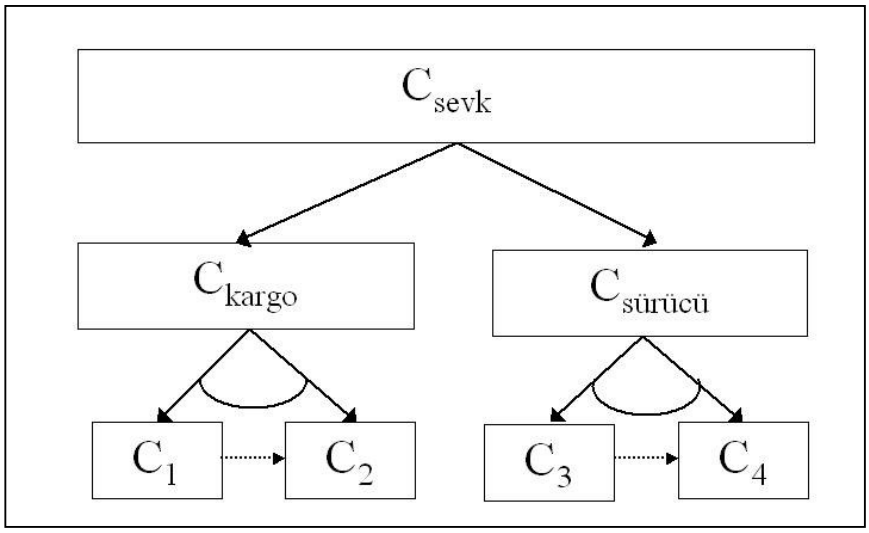
Ağaçtaki her düğüm, bir işi modelleyen bir taahhüt içerir. İş hayatında işler alt-işlere bölünebilir. Başka bir deyişle bir işi başarılı bir şekilde bitirebilmek için onu oluşturan alt-işleri bitirmek gerekir. Bu olayı da taahhütleri (yani olayları) alt-taahhütlere (yani alt-olaylara) bölerek modelliyoruz. Her alt-taahhüt yukarıda tanımlandığı gibi normal bir taahhüttür. Ancak alt-taahhütler bir araya gelerek üstteki ana-taahhüdü oluştururlar. Alt-taahhütler aralarında mantıksal VEYA (OR) ya da mantıksal VE (AND) işlemleriyle bağlı olabilirler. Bu yüzden VE/VEYA ağaçlarını alanı göstermek amacıyla kullanıyoruz. Sistem tanımlama safhasında, taahhütler VE/VEYA ağacını oluşturmak içi uygun şekilde girilmeli ve işaretlenmelidirler. Bu bölünme faaliyeti taahhütteki tüm borçlu ve alacaklı tarafların çoklu-etmenli sistemimizin sınırları içerisinde kaldığı sürece devam edebilir. Eğer taahhüdün başarılı ya da başarısız bir şekilde bitmesi algılanamayacaksa bu borçlu ve alacaklı taraflar için taahhüt tanımlanamaz. Böylece taban seviyesinde bulunan bir taahhüdün başarılı bitişini ya da başarısız bir bitişi ile kural dışı duruma yol açtığını algılayabiliyor ve dolayısıyla onu düzenleyecek hareket planı için gereken çıkarsamayı kurabiliyoruz.

VE/VEYA ağacını kullanmanın ana amacı, kural dışı durumları modellerken edimlerini yerine getirmeyen taahhütleri arayıp bulmak ve böylece uygulanması gereken doğru hareket planını uygulamaktır. Yukarıdaki örnekte eğer sevkıyat olmadıysa, düzeltici hareket olarak yeniden sevkıyat istemek yine aynı kural dışı duruma sebep olabilir. Fakat daha fazla araştırıp, kural dışı durumun sürücüden dolayı oluştuğunun bulunması, bu sefer sevkıyatı kargo şirketiyle yapmamızı sağlayacaktır (VEYA – OR bağlantısı) .

Sistem başlangıç kural konfigürasyonu ile çalışmaya başlar. Uygulama esnasında sistem her kural dışı durum oluşumunda paralel bir yapı yaratır ve VE/VEYA ağacının içinde arama yaparak tam olarak uyumsuzluk yapan düğümü bulur.

2.3. Kural Dışı Durumları İzlemek

Tüm taahhütleri kalıcı olarak zaman aşımı değerleriyle kayıt etmek olasıdır. Verimlilik ve doğruluk amacıyla, tüm taahhütleri merkezi bir veri havuzunda tutuyoruz. Bu veri havuzunu "izleyen etmen" (MA) olarak adlandırdığımız bir etmen yönetmektedir. MA tüm taahhütlerin zaman aşımı verip vermediğini denetler. Eğer bir taahhüt zaman aşımına uğradıysa, bu durum burçlu (sorumlu) tarafın edimini yerine getirmediği anlamına gelir, aksi takdirde MA konu ile ilgili bir ileti almalıydı. Bu durumda MA taahhüdün alacaklısına borçlunun başarısız olduğuna dair ileti gönderir. Eğer bu taahhüdün yerine getirilmemesi başka taahhütleri de yerine getirilemez kılıyorsa bu taahhütlerin de alacaklıları haberdar edilir.



Şekil 1. "Sevkıyat" senaryosu çerçevesinde kural dışı durumların modellenmesi

2.4. Örnek

Yukarıda resmedilmiş örneği varsayalım. Örnek çeşitli malların göndericiden alıcıya sevk edilmesi sürecini seçenekli yöntemlerle modellemektedir:

1. Aşağıdaki taahhüdün *Varış zamanı* zaman aşımı değeriyle yaratıldığını varsayalım.
   C_{sevk} =T *(Gönderici, Alıcı, Mal-Gönder, Varış zamanı)*
   Bu parametre başarılı bir biçimde görevin yerine getirilip getirilmediğini kontrol edecektir. Eğer taahhüt normal yollardan yerine getirilmezse, otomatik olarak Gönderici Mal-gönder yüklemini yerine getirmemiş sayılacaktır ki bu da bir kural dışı duruma yol açmaktadır.
2. Daha sonra MA (bkz Bölüm 2.3) C_{sevk} taahhüdünü yaratılmasının hemen ardından kaydeder.
3. MA *Alıcı* etmenden taahhüdün yerine getirilip getirilmediği ile ilgili bilgi ya da ileti bekler.
4. Eğer *Alıcı* etmenden uygun yanıt gelmez ve bu arada da *Varış zamanı* dolarsa, o zaman MA *Gönderici* etmenin edimini yerine getirmediğini anlar ve *Alıcı* etmeni bu sorundan haberdar eden bir ileti gönderir.
5. Edimin yerine getirilmediği durumu daha iyi anlayıp inceleyebilmek için C_{sevk} taahhüdü (ya da en son bilgisi gelen kural dışı durum) bağlantılı alt-taahhütlere bölünebilir.
   $C_{sevk} = C_1 \wedge C_2$ **öyle ki**
   C_1 =*T(Gönderici, KargoŞirketi, YüklemeYap,* t_1*)* ve
   C_2 =*T (KargoŞirketi, Alıcı, YükTaşı, VarışZamanı)* ve
   $t_1 < VarışZamanı$
6. Seçenekli olarak (Şekil 1'deki ana OR dallanması) aşağıdaki bölünmeyi de yapabiliriz.
   $C_{sevk} = C_3 \wedge C_4$ **öyle ki**
   C_3 =*T (Gönderici, Sürücü, YükVer,* t_1*)* **ve**
   C_4 =*T (Sürücü, Alıcı,YükTaşı, VarışZamanı)* **ve**
   $t_1 < VarışZamanı$
7. Son olarak Şekil 1'i daha iyi ve anlaşılır bir gösterime ulaşmak amacıyla çizdik. Her taahhüt bir alt-taahhüde, borçlu ve alacaklı etmenle etmen topluluğuna dâhilse, bölünebilir.
8. Önerilen yapının sonucu VE/VEYA ağacıdır. Bu ağaç ayrıntılarıyla [10]'da anlatılmaktadır. Bu yapıda yapraklar en son bölünebilen kural dışı durumlardır. Bu yapraklardaki kural dışı durumlardan düzeltici hareket planlarına ulaşmak mümkün olmaktadır.

2.5. VE/VEYA Ağacının İşlenmesi

Yapıya bir düğüm (taahhüt) eklemek için, eklenecek düğümün ağaçtaki seviyesini ve kardeş düğümlerle olan ilişkisini (AND ya da OR) bilmeliyiz. Gerisi genel bir "ağaca düğüm ekleme" algoritmasıdır. Düğüm silmek için, düğümün ağaçtaki tam yerini belirtmek yeterlidir. Sistem silinen düğümün kardeş düğümü olup olmadığını kontrol edecek, duruma göre alt-düğümü olmayan baba-düğümü yaprak-düğümü olarak işaretleyecektir. Ağaçta herhangi bir düğümün yerini değiştirmek ise önce eskiyi silip daha sonra yeniyi girmek şeklinde olacaktır.

Yaprak düğümü taban seviyesinde bir düğüm içerir. Etki alanı içerisinde daha fazla bölünme burada mümkün değildir. İstisnaları işlemek için gerekli tüm hareketler ve planlar yaprak düğümlerde bulunan taahhütlerden başlatılacaktır. Ağaç durağan olmayabilir. Sisteme her yeni düğüm (taahhüt) eklendiğinde istisnaların modeli değişecektir. Hangi durumlarda hangi düğümlerin eklendiği kodlanabilir.

3. Kural Dışı Durumları Belirlemek için Ontoloji Kullanımı

Bilinen bir ontoloji dili olan OWL kullanarak bir kural dışı durum ontolojisi geliştirdik [11]. Bu ontoloji kural dışı durumlar arasındaki benzerlikleri ve diğer ilişkileri yakalamasını ve böylece sistemin benzer kural dışı durumlar için benzer reçeteler hazırlamasını sağlar.. Bu ontolojinin bileşenleri Bölüm 5'tedir. Bu ontoloji kurallarla birleştirilerek çıkarımlar yapmaya elverişli hale gelir.

Çoklu-etmen sistemimize gelen girdi hangi taahhüdün başarısız olduğuna dair hiç bir bilgi içermeyebilir. Sistemin kullanıcıları kural dışı durumları belki de zaman aşımı olmadan önce bile algılama gereksinimi duyabilirler. Bu durumda eğer ontoloji yeterli olacak ve olası kural dışı durumları yakalayabilecek şekilde tasarlanmışsa bu ontoloji ile ilgili örnekler veren bir veri kümesi, taahhütlerle yakalanamayacak kural dışı durumları yakalamamızı sağlayabilir. Bu sonuca, ontolojide sunulan kurallar üzerinde yapılacak akıl yürütme (çıkarsama) yoluyla ulaşılacaktır. Bu durumda bazı taahhütlerin gelecekte ihlal edilmiş olacağını önceden kestirmek mümkün olacaktır. Bu sistemdeki diğer etmenler ya da ihlal edilecek taahhütteki alacaklı konumda bulunan taraf böylece sorunu giderici ya da düzeltici alternatif hareket planlarını çağırabilmek için zaman kazanacaktır ki bu da verimliliği ciddi ölçüde arttıracaktır.

Şekil-1'deki örnekle ilişkili bir çıkarsama ve sonucu aşağıda verilmiştir. Kamyonla sevkıyat yapılan bir durumu ele alalım. Bu durumda sevkıyat yolu üzerinde bulunan bir etmenden bir ileti ya da bilginin alacaklı etmen tarafından düzenli abonelik yoluyla alındığını varsayalım. Bu durumda yollarda kazalarla ilgili bilgiler kaza olduğu anda elektronik yoldan tüm abonelere ulaşmaktadır. Böylece MA etmeni (bkz Bölüm-3) bu durumu önceden belirlenmiş bir biçemde alabilmekte ve daha ilgili taahhüdün varış zamana sona ermeden sorun olabileceğinin çıkarsamasını yapabilecektir.

Çoklu-etmen sistemimize gelen bu bilgi, yükümüzü taşıyan sürücünün izlediği yolla ilgili olabilir. O zaman çıkarsamamız şöyle olacaktır:

Ontolojide tanımlı olgular:

- *Sevkıyatı sürücüler yapar.*
- *Sürücüler kamyon kullanır.*
- *Kamyonlar çalışır ya da durmuş halde olabilirler.*

Ontolojide tanımlı kural:

- *Eğer bir kamyon durmuş ise sürücüsü onu süremez.*
- *Bir sevkıyat, sürücüler ilgili kamyonu süremediklerinde iptal olur.*

Bu tip çıkarsamalar için çeşitli kural-tabanlı otomatik çıkarsama sistemleri, tanımlanmış ontolojiden üretilen nesneler üzerine uygulanarak elde edilebilir.

4. Tartışma ve Sonuç

Yukarıda uygulanan örnekler tedarik zinciriyle alakalı olmakla beraber geliştirilen yöntem geneldir ve başka uygulamalarda da kullanılabilir. Tabii ki uygulamaya bağlı olarak başka bir ontoloji kullanılması ya da geliştirilmesi gerekecektir. Fakat bunun dışında kullanılan, taahhüt ve VE/VEYA ağaçlarının yapılarında bir değişiklik olmayacaktır. Endüstri ile olan birebir bağlantısından dolayı tedarik zincirleri yazılım mühendisliğinde çok ilgi çeken bir alandır. Tedarik zinciri otomasyonunda ve tedarik zinciri kural dışı durumlarının işlenmesinde çeşitli yaklaşımlar izlenmiştir.

Huhns ve öbürleri tedarik zinciri yönetiminde gerekecek eşgüdümü modellemek için dilbilim örnekleri kullanmaktadırlar [5]. Öncelikle etkileşim şemaları geliştirdiler. Daha sonra "use case modeli" ile etmenler arası konuşmayı belirlediler. Oradan da Dooley çizgelerini elde ettiler. Son olarak da "sonlu otomat"ları inşa ederek gerekli etmen ana iskeletlerine ulaştılar. Bu sıra, olağan tedarik zinciri yönetimi için bir üretim zinciri oluşturmak gibi görünmektedir. Ancak kural dışı durum işlenmesi bu süreçte yalnızca dikkat atfedilen bir adım olarak kalmıştır. Yazarlar, kural dışı durum sınıflarını tanımlamışlar ancak kural dışı durumlarla başa çıkabilmek için sağladıkları katkı yetersiz kalmıştır zira taahhütler bir taksonomi çerçevesinde ayrıntılı olarak belirtilmemiştir. Hatta kavramları tanımlayacak ve kavram sözlüğü gibi kullanılabilecek ontoloji türünde bir araç da önerilmemiştir.

Fox ve öbürleri tedarik zincirlerini akıl yürütebilen ve bağımsız davranabilen akıllı yazılım etmenleriyle modeller [2]. Fox ve öbürlerine göre, etmen tabanlı bir yapı kullanılınca, tedarik zinciri yönetim sisteminin dinamizm, çıkarsama, yanıt vericilik, işbirliğine hazır ve geriye dönük uyumluluk içerme, etkileşimlilik ve uyarlanırlık özellikleri vardır. Kolayca görülmektedir ki, bu özellikler sadece geleneksel tedarik zinciri yönetimi için değil, aynı zamanda tedarik zincirlerinde oluşabilecek kural dışı durumlarda da geçerlidir. Fakat, Fox ve öbürleri tedarik zincirlerinin etmenlerle modellemenin ilerisinde, kural dışı durumları işlemek için bir yöntem geliştirmemişlerdir.

Dellarocas ve Klein [6] da dilbilim modeli yerine bir bilgi tabanı yaklaşımı önermektedirler. Bu çalışma kural dışı durumlarla ilgili çalışmalar için daha kuvvetli bir yaklaşım içermektedir. Taksonomi n-bacaklı bir ağaçtır. Ancak bu ağacın çocuk-düğümleri arasında herhangi bir ilişki düşünülmemiştir. Üstelik çocuk-düğümler (alt-taahhütlerin bulunduğu) ile baba-düğümler arasında da herhangi bir ilişki mevcut değildir.

Kalakota ve öbürleri tedarik zinciri mimarisini durağandan (statik) hareketliye (dinamik) taşımaktadırlar [12]. Bunu çoklu-etmen ortamı kullanarak yapmaktadırlar. Müşterilerden, perakendecilerden, dağıtım merkezlerinden ve üretim tesislerinden tarihsel verileri toplayarak, karar vermede kullanmak üzere bilgi tabanını bu toplanmış bilgilerden oluşacak durağan bir altyapı ile kullanmak yerine, tüm bilgiyi yerel etmenlerde hareketli (dinamik) olarak tutup daha küçük zaman aralıklarında daha sık ve hızlı kararlar vermeyi hedeflemişlerdir. İçsel işlemlerdeki varsayımları her etmenin kendi hakkında karar verme yetisine bağlı olarak istatistiksel metotlarla belirlenmektedir. Yazarlar, tedarik zinciri ile ilgili bir ontoloji de önermektedirler ancak kural dışı durum yönetimi için çok fazla önerileri yoktur.

Frey ve öbürleri [13] sisteme ek bir özellik önermişlerdir. Bu da çeşitli ve bağımsız çoklu-etmenli sistemin, üst seviyeli bir topluluk oluşturmak amacıyla tümleştirilmesidir. Yazarlar bize çoklu-etmen altyapısının küresel/esnek ve yerel/özerk iş ihtiyaçları için mükemmel olarak uygunluğunu göstermişlerdir. Bu sistemlerde rekabet ve birlikte çalışma kavramları birbirlerini rahatsız etmeden

birlikte bulunabilirler. Yazarlar zaten mevcut ya da geliştirme altında olan çoklu-etmenli projeler kullanmaktadırlar. Her çoklu-etmen sistemi, müzakere, süreç planlama, üretim planlama ve denetimi gibi farklı görev sınıflarından sorumludur. Yazarların çalışması bu alt sistemlerin arasındaki etkileşimi ve tümleştirmeyi tanımlar. Ancak tüm zamanlama, izleme, güvenilirlik inşa etme çabalarına rağmen kural dışı durumlar dikkate alınmamıştır. Bu da tedarik zinciri yönetiminde kural dışı durumlar için özel ilgi ve çabanın gerekliliğini ortaya koymuştur.

Sonuç olarak, yazılımlar arası kural dışı durumların otomatik olarak önlenebileceği ve düzeltici durumlarla ilgili hareket planlarına ulaşılabileceği gösterilmiştir. Modelimiz karma bir metot kullanarak ya VE/VEYA ağaçlarındaki taahhütlerin ihlallerinden kural dışı durumları saptamakta ya da daha bu taahhütlerin zaman aşımı bilgisi gelmeden ontoloji yardımıyla sorun çıkacağını önceden kestirerek büyük ölçüde verimlilik sağlamaktadır.

Burada açıkladığımız metot gelişmelere açıktır. Eğer sistem küresel ve dev bir ağa doğru daha büyük ve karmaşık olacak şekilde büyüyecek ve tedarik zinciri dünyasındaki tüm etmenleri içerecekse, tüm taahhüt verisini tek bir MA içine yükleme zor ve yapılabilirlikten uzak olacaktır. İletişim ve depolama kısıtları sorun yaratabilecektir. Bu durumda MA etmeninin dağıtık bir versiyonu başarım ve özellik hedeflerinden dolayı tasarlanabilir. Benzer şekilde VE/VEYA ağacı etmenler arasında dağıtık olarak tutulabilir.

5. Ek: Kural Dışı Durumları Bulmak İçin Kullanılan Ontoloji

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY owl 'http://www.w3.org/2002/07/owl#'>
]>
<rdf:RDF
    xml:base="http://boun.edu.tr/driver"
    xmlns:a="http://boun.edu.tr/driver#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:swrl="http://www.w3.org/2003/11/swrl#">
<owl:Ontology rdf:about=""/>
<owl:ObjectProperty rdf:ID="cancelled">
    <rdfs:domain rdf:resource="#delivery"/>
    <rdfs:range rdf:resource="#truck"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="doneBy">
    <rdfs:domain rdf:resource="#delivery"/>
    <rdfs:range rdf:resource="#driver"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="drives">
    <rdfs:domain rdf:resource="#driver"/>
    <rdfs:range rdf:resource="#truck"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="workMode">
    <rdfs:domain rdf:resource="#truck"/>
</owl:ObjectProperty>
<a:driver rdf:ID="George">
    <a:drives rdf:resource="#bmc"/>
</a:driver>
<a:driver rdf:ID="John">
    <a:drives rdf:resource="#fiat50NC"/>
```

```
</a:driver>
<a:truck rdf:ID="bmc">
    <a:workMode rdf:resource="#working"/>
</a:truck>
<a:delivery rdf:ID="del01">
    <a:doneBy rdf:resource="#John"/>
</a:delivery>
<a:delivery rdf:ID="del02">
    <a:doneBy rdf:resource="#George"/>
</a:delivery>
<a:delivery rdf:ID="del03">
    <a:doneBy rdf:resource="#John"/>
</a:delivery>
<a:delivery rdf:ID="del04">
    <a:doneBy rdf:resource="#George"/>
</a:delivery>
<a:truck rdf:ID="fiat50NC">
    <a:workMode rdf:resource="#stopped"/>
</a:truck>
<swrl:Variable rdf:ID="V"/>
<swrl:Variable rdf:ID="T"/>
<swrl:Variable rdf:ID="D"/>
<swrl:Imp>
    <swrl:head rdf:parseType="Collection">
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#cancelled"/>
            <swrl:argument1 rdf:resource="#V"/>
            <swrl:argument2 rdf:resource="#T"/>
        </swrl:IndividualPropertyAtom>
    </swrl:head>
    <swrl:body rdf:parseType="Collection">
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#doneBy"/>
            <swrl:argument1 rdf:resource="#V"/>
            <swrl:argument2 rdf:resource="#D"/>
        </swrl:IndividualPropertyAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#drives"/>
            <swrl:argument1 rdf:resource="#D"/>
            <swrl:argument2 rdf:resource="#T"/>
        </swrl:IndividualPropertyAtom>
        <swrl:IndividualPropertyAtom>
            <swrl:propertyPredicate rdf:resource="#workMode"/>
            <swrl:argument1 rdf:resource="#T"/>
            <swrl:argument2 rdf:resource="#stopped"/>
        </swrl:IndividualPropertyAtom>
    </swrl:body>
</swrl:Imp>
<owl:Class rdf:ID="delivery"/>
<owl:Class rdf:ID="driver"/>
<owl:Class rdf:ID="truck"/>
</rdf:RDF>
```

Kaynakça

[1] H. van Vliet. *Software Engineering: Principles and Practice*, Second Edition, Wiley, 2000.

[2] M. S. Fox, M. Barbuceanu ve R. Teigen. *Agent-oriented supply-chain management*. The International Journal of Flexible Manufacturing Systems, 12(2000) s. 165-188.

[3] N .Jennings. *On Agent-Oriented Software Engineering*. Artificial Intelligence Journal, 117 (2) 277-296, 2000.

[4] T. J. Becker, "*Putting a Price on Supply Chain Problems: Study Links Supply Chain Glitches with Falling Stock Prices*", Georgia Tech Research News, Aralık 2000, Web adresi: http://www.gtresearchnews.gatech.edu/newsrelease/

[5] M. N. Huhns, L M.Stephens ve N Ivezic. *Automating supply-chain management*. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), s. 1017-1024. ACM Press, Temmuz 2002.

[6] C. Dellarocas ve M. Klein. *A knowledge-based approach for designing robust business processes*. W. van der Aalst et al. (eds.): Business Process Management, LNCS 1806, s. 60-65, 2000.

[7] C. Dellarocas, M. Klein ve J. A. Rodriguez-Aguilar. *An exception handling architecture for open electronic marketplaces of contract net software agents*. Proc. of the ACM Conf. on Electronic Commerce, s. 225-232, 2000.

[8] N. Fornara ve M. Colombetti. *A commitment-based approach to agent communication.* Applied Artificial Intelligence: an International Journal, 18(9-10):853-866, 2004.

[9] P. Yolum ve M. P. Singh. *Reasoning about commitments in the event calculus: An approach for specifying and executing protocols*. Annals of Mathematics and Artificial Intelligence 42:227-253, Kluwer Academic Publishers, 2004.

[10] N. J. Nilsson, *Principles of Artificial Intelligence*, Springer Verlag, 1980.

[11] OWL. *Web Ontology Language*. Web adresi : http://www.w3.org/TR/owl-features/

[12] R. Kalakota, J. Stallaert ve A. B. Whinston, *Implementing Real-time Supply Chain Optimization*, Web adresi: http://cism.nccombs.utexas.edu/jan/sc_imp.html

[13] D. Frey, T. Stockheim, P. O. Wolk ve R. Zimmermann. *Integrated multiagent based supply chain management*. Proceedings of the First International Workshop on Agent-based Computing for Enterprise Collaboration (ACEC), 2003.