DEU FMD 22(64), 219-231, 2020

Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi
Dokuz Eylul University Faculty of Engineering Journal of Science and Engineering

Basılı/Printed ISSN: 1302-9304. Elektronik/Online ISSN: 2547-958X

İş Akış Motoru Tasarımı ve Gerçekleştirilmesi

Workflow Engine Design and Implementation

Oğuzhan Kayış [1*], Semih Utku [1]

[1] Dokuz Eylül Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği Bölümü, İzmir, TÜRKİYE
Sorumlu Yazar / Corresponding Author *: oguzhan.kayis@ceng.deu.edu.tr

Geliş Tarihi / Received: 27.06.2019 *Araştırma Makalesi/Research Article*
Kabul Tarihi / Accepted: 08.08.2019 *DOI:10.21205/deufmd.2020226422*

Atıf şekli/ How to cite: KAYIŞ, O., UTKU, S. (2020). İş Akış Motoru Tasarımı ve Gerçekleştirilmesi. DEUFMD 22(64), 219-231.

Öz

Bu çalışma özellikle büyük ölçekli işletmelerin süreçlerinde kullandıkları bir yapı olan iş akışı motorunun dinamik işleyecek şekilde bir tasarımını ve gerçekleştirilmesini kapsamaktadır. Üzerinde çalıştığımız temel konu, geliştirdiğimiz sistemin farklı işletmelere ve işletmeler içerisindeki farklı süreçlere uyum sağlayabilecek esneklikte olmasıdır. Endüsriyel alanlarda iş süreci bir çok etmene göre zamanla değişmektedir. Bu amaçla çalışma kapsamında iş akış motoruyla bütünleşik çalışabilen bir kural motoru geliştirilmiştir. Kural motoru, iş akış motorunun değişen iş süreçlerine çalışma zamanlı olarak uyum sağlamasını amaçlamaktadır. Tasarlanan bu sistem, bir teknoloji ürünleri firmasında, ürünlere bağlı belgelerin takip süreçlerini otomatize etmek üzere uygulanmıştır. Uygulama sonucunda toplanan bilgiler, sıralı örüntü madenciliği yöntemleri kullanılarak analiz edilmiştir.

Anahtar Kelimeler: *İş Akış Motoru, Kural Motoru, Sıralı Örüntü Madenciliği, Belge Takip Sistemi*

Abstract

This study includes a design and implementation of a workflow engine which is a structure that is used in the processes dynamically of the large-scale enterprises. The main issue is creating a dynamic system that is flexible enough to adapt to different businesses and different processes within the enterprises. Business process in industrial areas changes over time according to many factors. For this purpose, within the scope of the study, a rule engine integrated with the workflow engine was developed. The rule engine enables the workflow engine to adapt to changing business processes at runtime. This system was designed to automate the follow-up processes of the documents connected to the products in a technology products company. In addition, the results of the study were analysed using sequential pattern mining methods.

Keywords: *Workflow Engine, Rule Engine, Sequential pattern mining, Document Tracking System*

1. Giriş

20.yy'ın sonlarına doğru firmalar, bünyelerine başka firmaları almaya veya çeşitli alt birimlere bölünmeye başladılar. Farklı birimlerden oluşan firmalar grubu, kendi içlerindeki işlerde de satış, lojistik ve dağıtım, pazarlama, üretim, finans gibi alt modüllere ayrıldılar. Büyüyen firmaların alt bölümleri arasındaki iletişim ve sistematik gereksinimi bilgi teknolojileri birimlerinin oluşmasını ve ERP çözümlerini gerektirdi [1]. Gelişen firmaların bünyesinde kurulan bilgi ve teknoloji birimleri yazılım temelli iletişim çözümleri üretmektedir. Elektronik posta alt yapısı ile beslenen bu çözümler farklı iş akış süreçlerini üretmiş ve bu süreçlerin yönetilebilmesi için de bu modüllerle bütünleşmiş çalışabilen iş akış motorları geliştirilmiştir.

İş akışı motoru, endüstriyel anlamda iş süreçlerini yöneten içerisinde yazılım ve insan kaynakları barındıran bir sistemdir [2]. Firmalar iş akış motorlarını, özellikle satış & dağıtım ve muhasebe bölümlerince sürece dahil olan satış siparişi, satın alma siparişi, fatura, irsaliye gibi belgelerin takibinde kullansalar da üretim sahasında da ürün ile ilgili bilgilerin birimler arasında paylaşımı amacıyla da kullanmaktadırlar.

Endüstriyel anlamda iş genellikle birden fazla aktiviteden oluşmaktadır. Her bir aktivite belirli bir işlem birimi tarafından gerçekleştirilir. Bu birmlere örnek olarak çalışan gruplar, yazılım ya da veri tabanı yönetim sistemleri verilebilir [3]. İş akışı sistemleri de iki farklı grup altında toplanmaktadır. Bunlardan ilki faaliyet tabanlı olan iş akışı yönetim modelidir. Bu modelde işin tamamlanması için gereken faaliyetlere odaklanılır. Diğer bir tür ise obje (varlık) tabanlı uygulamadır. Bu modelde ise ilkinin aksine işin tamamlanması için objenin mümkün olan hareketleri tanımlanır [4]. Bu çalışma obje tanımlı bir iş akışı motoru geliştirilmek üzere tasarlanmıştır.

Bu çalışma kapsamında iş akışı motoru modellemesi aşamasında sonlu durum makineleri yapısı kullanılmıştır. Bu çalışma bir sonlu durum makinesinin ilişkisel bir veri tabanında nasıl saklanabileceği sorusuna da cevap vermektedir. Ayrıca çalışma kapsamında iş akışı motorunu esneklik yönüyle güçlendiren bir kural motoru da tasarlanmış ve gerçekleştirilmiştir. Kural motoru doğrudan yazılımcı olmayan bireylerin, iş süreci yönetim sistemindeki akışa müdahale edebilmesine olanak sağlayan bir yazılım bileşenidir. Son yıllarda kolay okunabilir ve bakım yapılabilir bir yöntem olduğundan oldukça popüler bir hale gelmiştir [5,6]. İş kuralı bir akışı ve prosedürünü belirlemeye yarayan ifadedir. Geliştirilen kural motoru ile temel bir yazılım diline benzer şekilde kullanıcı ifadeleri karşılaştıracağı, sonucu doğru ya da yanlış olabilen denklemler kurabilir. Bu denklemlerin sonucuna göre atamalar yapabilir.

Tasarlanan bu çözüm bir teknoloji üretim firmasında ürünlerle ilgili belgelerin takibi süreçlerini sanallaştırmak amacıyla gerçekleştirilmiştir. Firmanın dış ticaret bölümünün gerçekleştirdiği belge talebi ile başlayan süreç, talebi oluşturulan belgenin önceden sınıflandırılan birimlere iletilmesi, belgelerin onay, ret, hazırlık ve talep sahibine geri bildirim adımlarını kapsayacak şekilde uygulanmıştır. Bu sistemden kurulmasından önce, firmada manuel olarak yönetilen bu süreç, bu çalışmayla birlikte otomatize edilmiştir. Geliştirilen sistemin içerisinde kullanıcı işlem bazlı veriler loglanmıştır. Projenin canlı sürece geçmesiyle birlikte ilk altı ayın sonucunda, girilen belge taleplerinden elde edilen veri üzerinde, sıralı örüntü madenciliği teknikleri uygulanmıştır. Alınan sonuçlar ilgili iş birimleri ile paylaşılmış ve talep edilen bazı belgelerin bir paket halinde talep edilmesi önerilmiştir.

1.1. İlişkili çalışmalar

Literatürde, yapılan çalışmaları iş akış ve kural motoru üzerinden iki farklı şekilde işlendiği görülmüştür. Bu sistemler genellikle birbirinden bağımsız olarak geliştirilmiş olsalar da özellikle gelişen ERP (Entireprice Resource Planning) sistemleri ile bu iki ürünün bütünleşik yapıda kullanıldığı durumlar da bulunmaktadır.

Bir grup bilim adamının 1995 yılında yayınladıkları bir makalesinde gelişen endüstriyel hayatın bir doğal sonucu ve gerekliliği olarak, iş akışı yönetim süreçlerini ele almıştır. Çalışmada iş akışının modellenmesinden uygulanmasına kadar olan süreçteki farklı yaklaşımlar sınıflandırılmış ve iş akışı yönetiminde kullanılan terimler açıklamıştır [7].

Wil van der ve Aalst Kees van Hee, 2000 yılında yayımladıkları kitapta iş akışı süreçleri, yönetimi ve bu sürecin bilgi teknolojileri ile ilişkisini derinlemesine incelemişlerdir. Bu çalışmada sunulan modelde petrinetler üzerinden bir modelleme tarif edilmektedir. Kitapta yapılan çalışmanın uygulanabileceği alanlar bir seyahat acentesi, bisiklet fabrikası, sürüş okulu gibi farklı endüstri kolları üzerinde örneklendirilmiştir. Çalışmada iş akışının işletmenin verimini artırabilmesi için sürdürülebilir ve okunabilir bir yapıda tasarlanması gerektiği vurgulanmaktadır [8].

Chun Ouyang ve diğerlerinin 2010 yılında yayımladıkları bir diğer çalışmada, daha eski bir çalışmaya atıfta bulunarak petrinetlerin bir alt kümesi olarak WF-net'lerden bahsettikten sonra YAWL isimli projeyi farklı başlıklar altında incelemişlerdir. Bir iş akış motorunun süreci nasıl tanımlayabileceği gibi genel kavramlardan da bahsedilen bu çalışmada, bizim çalışmamıza benzer olarak ayrıca veri madenciliği üzerinde de durulmuştur. Hem sistem hem de işlem loglarının anlamlı bir veri içerebileceğinden ve bu veri üzerinde yapılacak bir çalışmanın faydalı olabileceğinden bahsedilmiştir [2].

Bouafia ve Molnár 2018 yılında yayınladıkları çalışmalarında iş akışı modellemesi için kullanılan yapıları statik ve dinamik olarak ikiye ayırmıştır. Makalede günlük hayatta endüstrinin ihtiyaçlarının dinamik olarak değiştiği ancak sıklıkla kullanılan iş akışı motorlarının statik yapıda olduğu vurgulanmış ve dinamik iş akışı motoru için öneride bulunulmuştur. Ayrıca bu çalışmada iş akış motoru başlığı altında ayrıca açıkladığımız temel desenler de açıklanmıştır [9].

İş akış motorlarının evriminin anlatıldığı Krasimira ve diğerlerinin yapmış olduğu çalışmada; iş akış motorlarının evrimsel gelişimi başlığı altında hem tarihsel süreç incelenmiş hem de iş akış motorları üç farklı şekliyle sınıflandırılmıştır. Bu başlıklar; ticari ve bilimsel anlamda yazılımlar, yazılım diline göre araçlar, destekledikleri standartlara göre araçlar şeklinde sınıflanmıştır[10].

Çalışmalar kapsamında Erwin ve Jacops'ın bir kural motoru da geliştirildiği görülmektedir. Erwin ve Jacops'ın geliştirdikleri Java tabanlı kural motoru ile ilgili makalelerinde, kural motorunun birçok alanda kullanabilecek bir yazılım olmasına karşın yaygın olarak kullanmadığı vurgulanmıştır [11]. Geliştirilen çözümün herhangi bir Java uygulaması ile bütünleşmiş çalışabildiğinden bahseden yazarlar ayrıca kural motorunun bir bildirim aracı olarak da değerlendirilebileceğinin üzerinde durmuşlardır.

Rakesh ve diğerlerinin yaptığı çalışmada graph modeli üzerinden incelenen örnek iş akışı şemalarının logları üzerinde durulmuştur. Hem gerçek hem de örneklemeler üzerinden oluşturulan sentetik veriler üzerinden, çeşitli analizler yapılarak bir iş akışının başlangıç ve bitiş noktaları sıralanmış ve veri kümeleri oluşturulmuştur. Oluşturulan veri kümeleri üzerinden graph modeli üzerindeki en kısa yol bulunması ya da gürültü (noise) verinin belirlenmesi çalışmaları uygulanmıştır [12].

İş akış motoru uygulamaları ile veri madenciliği yöntemlerinin kullanıldığı bir diğer uygulamada da ilk olarak 2003 yılında Aalst ve diğerleri tarafından yayınlanmıştır. Çalışmada iş akışı motorları üzerindeki veri madenciliğinin başlangıç noktası, log olarak tanımlanmıştır. İşlemsel (transactional) log verileri üzerinden anlamlı verinin nasıl çıkarılabileceğinin örnekleri verilen bu çalışmada ayrıca ERP, CRM (Customer Relationship Management) uygulamalarında da benzer şekilde işlemsel log verisi üzerinden örnekler verilmiştir [13].

Singh ve Sandeep'in 2011 yılında yayınladıkları bir çalışmada yine iş akışı motorlarını üzerinden veri madenciliği yöntemleri üzerinde durulmuştur. Ancak diğer çalışmalarda loglardan anlamlı veri çıkarmak üzere iken bu çalışmada amaç; eylemleri inceleyerek petrinetler ile yeni bir model tasarlanmasıdır [14]. İş akışı yönetim sistemlerinde modelin belirlenmesi, iş ekibi açısından yüksek bir sorumluluk ve zaman gerektirdiğinden bu işlem için veri madenciliği kullanılması önerilmiştir.

Bu çalışma özünde bir iş akış motoru tasarımı ve gerçekleştirilmesine içerse de arka planda bu iş akış motoruna esneklik kazandıracak ek bir kural motoru tasarımı sunmaktadır. Ayrıca yapılan çalışmanın örnek belge takip sistemi gerçekleştirmesinden elde edilen bilgiler ile veri madenciliği yöntemleriyle analizler yapılmıştır. Bu yönüyle sıralı örüntü madenciliğinin bir iş akışı motoru üzerinde nasıl uygulanabileceği de bu çalışma kapsamında gerçekleştirilmiştir.

2. Materyal ve Metot

2.1. İş akış motoru genel özellikleri

Yapılan bir başka çalışma kapsamında gerçekleştirilen iş akış motoru endüstriyel anlamda bir iş sürecinin tamamının ya da bir bölümünün otomatize edilmesi kurallarını kapsamaktadır [15]. Bu kurallar bir sonlu durum makinesi yapısına benzerlik gösterdiğinden çalışma kapsamında bir sonlu durum makinesinin ilişkisel bir veri tabanında nasıl modellenebileceğinin de üzerinde durulmuştur.

Temel bir akış içerisinde aşağıdaki yapılar bulunmaktadır [4];

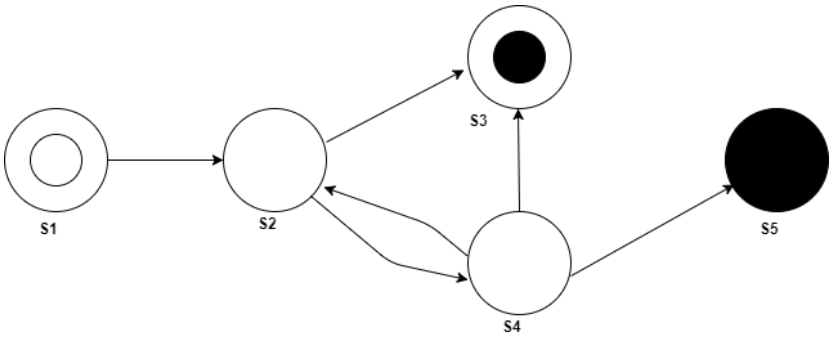
• Durumlar: takibi yapılacak objenin bir *t* anında bulunması mümkün olan her yapı bir durum olarak tanımlanır. Başlangıç, bitiş, iptal gibi özel durumların yanı sıra sayısız olarak uygulanan proje bazlı özelleşmiş durumlar olabilir.

• İşlemler: bir durumdan diğerine geçilmesi işlemidir. İş yapısı gereği genellikle tüm durumlar arasında bir bağ bulunmak zorunlu değildir. Başlangıç durumundan bir diğer duruma ve bu şekilde son duruma kadar olan tüm geçişler birer işlem olarak adlandırılır.

• Eylemler: işlemin gerçekleştirilebilmesi için gereken işleme eylem adı verilir. Bir iş akış motorunda eylemler mutlaka önceden belirlenmeli ve hangi eylemle hangi işlemin tamamlanacağına karar verilmelidir.

• Yetkiler: durumlar arasında geçişin anlatıldığı işlemler eylemler ile tetiklenir. Bu eylemlerin tetiklenmesi iş sürecinin güvenliği ve birbirinden farklı işlerin sahiplerinin ayrımı için gereklidir.

Şekil 1'de iş akış motorunu uyguladığımız belge takip sistemine ait bir belgenin akış diyagramı gösterilmektedir.



Şekil 1. Örnek bir akış diyagramı

Şekil 1'de gösterilen iş akış diyagramında S1, S2, S3, S4, S5 durumları temsil etmektedir. Örnek iş akış diyagramındaki verilen durumlar kategorilere ayrılmıştır.

• S1: Başlangıç durumu

• S2, S4: Geçiş durumu

• S3: İptal durumu

• S4: Bitiş durumu.

Takip edilen obje bir *t* anında bu yukarıdaki durumların herhangi birinde bulunabilir. Örnek gerçekleştirdiğimiz akışta kararlı sonlu durum makineleri ile modelleme yapıldığından bir nesne bir *t* anında sadece bir durumda bulunabilir [16].

Durumlar arasında gösterilen oklar işlemleri temsil etmektedir. Sistemde her bir işlem için bir ID tanımlanmıştır. Bu ID'ler işlemler için geçerli yetkilerin oluşturulması için kullanılacaktır. Arka planda tekrarsız olarak bir anahtarla belirlenen işlemler de belge takibi özelinde kategorilere ayrılmıştır. Ön yüzde belirlenen butonlar tetiklendiğinde ilgili tipteki işlem tetiklenmektedir.

• S1-S2: İstek yaratılması

• S2-S4, S4-S2: Sonraki/önceki adıma geçiş/onay

• S2-S3, S4-S3: İptal durumuna geçiş

• S4-S5: Doküman eklenmesi, final durumuna geçiş.

Bu işlemlerin gerçekleşmesi için gereken aksiyon eylemleri göstermektedir. Geliştirilen yapıda eylemler de ön yüzde tetiklenebilmesi için sınıflandırılmıştır. Aşağıda, durumlar arasındaki geçiş için tanımlanan aksiyon tipleri listelenmiştir. Bu aksiyon tipleri, iş akışında bir grup için tanımlanabileceği gibi ayrı ayrı gruplara da tanımlanabilir.

- İstek açma aksiyonu
- Sonraki adıma geçiş aksiyonu
- İsteği geri çekme aksiyonu
- İstek için belge eklenmesi, isteğin sonlandırılması aksiyonu.

Nihayetinde bu eylemler için gereken kontroller de yetki yapısını oluşturmaktadır. Geliştirilen sistemde yetki aksiyon ve yetkili grup beraberinde kullanılmıştır.

Klasik bir sonlu durum makinesini bir matris yapısında tutmak mümkündür [17]. Ancak eylemler durumlar ve işlemleri içeren bir yapı için tekrarsız ID'ler gerektiğinden ilişkisel bir veri tabanı oluşturulmuştur. Aşağıdaki Şekil-2 deki database diyagramda (ER Diagram) "_" işareti ile başlayan tablolar iş akış motorunun örnek gerçekleştirmesi olan belge takip sisteminin objeleri için olup, diğer tablolar genel yapıya aittir.

Şekil 2'de gösterilen tabloların içerikleri aşağıda açıklanmıştır.

State_TP: Durum tiplerini temsil eder, durumları içeren tablo ile ilişkilidir

States: Durumları temsil eder. Durum tiplerinin akış özelinde tekrarlanması amacıyla oluşturulmuştur.

Action_TP: Eylem tiplerini temsil eder, eylemleri içeren tablo ile ilişkilidir.

Actions: Eylemleri temsil eder. Eylem tiplerinin akış özelinde tekrarlanması amacıyla oluşturulmuştur.

_Doc_TP: Doküman takip sistemi içerisindeki her bir tekrarsız akış, belge tipi anahtar olarak belirlenmiştir.

_Process: Doküman takip sistemindeki her bir akışı temsil eder. Her bir akış bu sistem özelinde talep edilecek malzemenin tipi ve belgenin tipi ile özelleştirilmiştir.

Transaction: Bir akış içerisindeki tüm adımları göstermektedir. Burada bir durumdan geçilebilecek diğer durum da gösterilmektedir. Her bir durum için ayrı bir ID gösterildiğinden bir durumdan birden fazla duruma geçiş de tanımlanabilir.

User_Group: Sistem kullanıcı adı ile birimlerin eşlenmesi için kullanılmıştır.

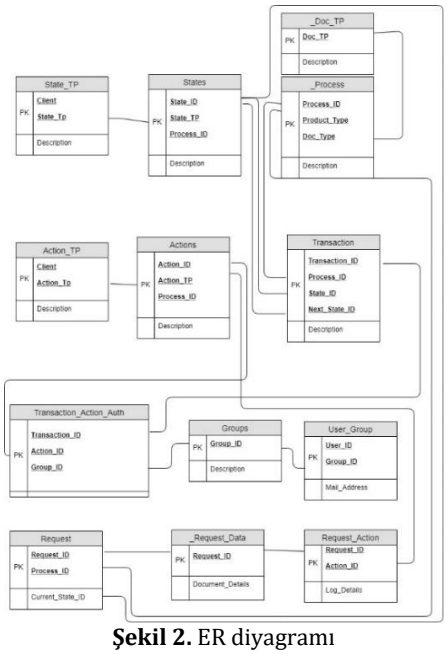
Groups: Yetkilerde kullanılmak üzere birimlerin tanımlandığı tablodur.

Transaction_Action_Auth: Her bir işlem için hangi eylemin geçerli olduğunu ve bu eylem için hangi grubun yetkili olduğu tanımlamak için kullanılmıştır.

Request: Her bir işlem (Process) için gerçekleştirilebilecek bir istek objesini temsil eder.

_Request_Data: Belge takibi işlemi için oluşturulan her bir isteği temsil eder. Bu tabloda belge takibi özelindeki malzeme numarası, isteği açanın mail adresi gibi detay bilgiler de saklanmaktadır.

Request_Action: İşlemsel log bilgilerinin saklanması için kullanılmaktadır. Tarih saat ve kullanıcı bilgileri ile işlem bilgilerini saklar. Örüntü madenciliği için temelde bu tablodaki bilgiler kullanılmıştır.



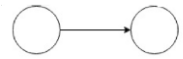
Şekil 2. ER diyagramı

2.2. İş akışı örgüleri

Bir iş akış moturunda, süreçlerin çalışmamızda modellediğimiz ve kullandığımız yapı gereği modelleyemediğimiz temel kontrol akış kalıpları aşağıda belirtilmiştir. Bu akış kalıpları genel olarak iş akış motorlarının özelliklerini ifade etmektedir [4].

2.2.1. Dizi (Sequence)

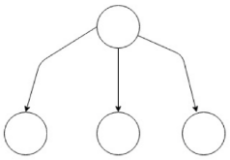
Dizi iş akış modeli, herhangi bir koşula bağlı olmaksızın doğrusal olarak bir durumdan diğer duruma geçişi temsil eden en temel kalıptır. Grafiksel gösterim Şekil 3'de belirtilmiştir.



Şekil 3. Dizi akış modeli

2.2.2.Paralel bölünme (Parallel split AND split)

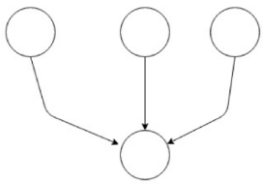
Paralel bölünme iş akış kalıbı bir akışı böler. Bu yapıda koşulsuz olarak bir akışın paralel olarak birden çok duruma bölündüğü durumu temsil eder. Grafiksel gösterim Şekil 4'te verilmiştir.



Şekil 4. Paralel bölünme modeli

2.2.3.Senkronizasyon (Synchronization AND-join)

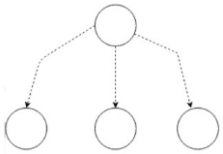
Senkronizasyon iş akış modeli, birden çok paralel durumun bir tek duruma eşitlenmesini ifade eder. Bu modelde paralel adımlar tam olarak bir adet durumda birleştirilirler. Grafiksel gösterim Şekil 5'de gösterilmiştir.



Şekil 5. Senkranizasyon modeli

2.2.4. Özel seçim (Exclusive choice XOR-split)

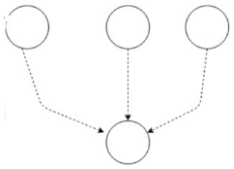
Özel seçim iş akışı kalıbı birden fazla yolu tanımlar. Bu model içerisinde bir obje birden fazla durum içerisinde bulunabilir. Bu çalışma kapsamında kullanılan kararlı durum makinesi bu yapıyı desteklemediğinden bu kalıba uygun olarak geliştirilmemiştir. Grafiksel gösterim Şekil 6'da verilmiştir.



Şekil 6. Özel seçim modeli

2.2.5. Basit birleştirme (Simple merge XOR-join)

Basit birleştirme modeli tanımlanmış olası yolları birleştirmek için kullanılır. Tam olarak bir durum için birleştirme tanımlanmıştır. Özel seçim modeline benzer olarak bu yapı da bu çalışma kapsamında geliştirilmemiştir. Grafiksel gösterim Şekil 7'de belirtilmiştir.



Şekil 7. Basit birleştirme modeli

2.2.6. Diskriminator (Discriminator)

Bu model varsayımlar işler için geliştirilmiştir. Özel seçim ile paralel bölünmenin ortak olarak kullanıldığı bir durumdur. Birden fazla durumun bir duruma bağlandığı ancak objenin bir sonraki duruma geçmesi için gelen tüm durumların tamamlanması gerektiği durumdur. Bir nesne durumu aktivite eder ve diğer durumlardan gelen eylemler de aktivasyonu tamamlar.

Bu çalışma kapsamında belge takip sistemi özelinde bu model geliştirilmemiş olsa da önerilen yapı bu modelin uygulanması için elverişli şekilde tasarlanmıştır.

2.2.7. Keyfi döngüler (Arbitrary cycles)

İş akış modellerinin genel bir yapısı da içlerinde döngülere izin vermemesidir. İş akış motorlarında bir veya daha fazla etkinliğin tekrarlanması desteklenmez. Bu çalışma kapsamında döngülerin engellenmesi için ek bir çalışma yapılmasa da uygulanan modellerde döngüler gerçekleştirilmemiştir.

2.2.8.Kesin sonlandırma (Implicit termination)

İş akışının sonlandırılması genel anlamda objenin geldiği son durumda yeni bir aktiviteye izin verilmediği durum olarak tanımlanmıştır. Ancak bazı modellerde işlemin doğrudan sonlandırılması da mümkündür. Bu çalışma kapsamında belge talebi açan kullanıcıların açtıkları talepleri hangi durumda olduğuna bakmaksızın iptal durumuna atayabilecekleri altyapı oluşturulmuştur.

2.2.9.İptal durumu (Cancel case)

Bir akış modelinin iptal edilmesi için geliştirilen modeldir. Bu çalışma kapsamında da iptal tipindeki durum tasarlanmıştır.

Tarif edilen modellerin diğer iş akışı sistemleri ile karşılaştırılması Tablo 1'de verilmiştir.

Tablo 1. İş akış motorlarının karşılaştırılması

İş akışı örgüleri	ezcworkflow[19]	YAWL[20]	ez publish 3[21]	GALAXIA[22]	RADICORE[23]	visual workflow[24]	verve workflow[25]	staffware[26]	mqseries workflow[27]	forté conductor[28]	hp changeengine[29]	fujitsu i-flow[30]	sap r/3 workflow[31]	Geliştirilen İş Akış Motoru
Dizi	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Paralel bölünme	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Senkranizasyon	√	√		√	√	√	√	√	√	√	√	√	√	
Özel seçim	√	√		√	√	√	√	√	√	√	√	√	√	
Diskriminator	√	√					√			√	√		√	√
Çoklu obje		√				√	√	√	√	√	√	√	√	
Keyfi döngüler	√	√			√		√	√		√	√		√	√
Kesin sonlandirma	√	√			√		√	√		√	√		√	√
Iptal durumu	√	√				√	√			√	√		√	√
Iptal eylemi		√						√						√
Kural motoru														√

2.3.Kural motoru

Kural motoru iş akış motorundan bağımsız da çalışabilen ve sistemde yazılım temelli olmayan kimselerin de koşullu kurallar yaratarak süreci çalışma zamanlı olarak değiştirebilmelerine olanak sağlayan bir yapıdır. İçerisinde bir yorumlayıcı ve semantik kontrol bileşeni barındırır. Önceden belirlenen değişkenlerin girilen koşullara göre değişmesine olanak sağlayan bu yapı, semantik anlamda üçüncü seviye programlama dillerine benzer özellikte geliştirilmiştir. Arka planda hazır olan değişkenleri kullandığından veri tipi tanımlamalarını içermez.

Bir kural motorunun en temel özelliği değişen iş süreçlerine çalışma zamanlı olarak uyum sağlayabilecek altyapıda tasarlanmasıdır [32].

Tablo 2. Operant listesi.

Operant	İsim	Açıklama
EQ =	Eşitlik	Eğer operant1 ve operant2 eşitse doğru olarak sonuçlanır.
NE != <>	Eşitsizlik	Eğer operant1 ve operant2 farklı ise doğru olarak sonuçlanır.
GT >	Büyüktür	Eğer operant1 operant2'den büyükse doğru olarak sonuçlanır.
GE	Büyük ya da eşittir	Eğer operant1 operant2'den büyük ya da eşitse doğru olarak sonuçlanır.
LT <	Küçüktür	Eğer operant1 operant2'den küçükse doğru olarak sonuçlanır.
LE	Küçük ya da eşittir	Eğer operant1 operant2'den küçük ya da eşitse doğru olarak sonuçlanır.
CO	İçerir	Eğer operant1 operant2'deki karakterleri içeriliyorsa doğru olarak sonuçlanır.
CN	İçermez	CO işleminin tersi durumdur.
CA	İçerir (En az 1)	Eğer operant2 operant1'den en az 1 karakter içeriyorsa doğru olarak sonuçlanır.
NA	İçermez (En az 1)	CA işleminin tersi durumdur.
CS	Karakter içerir	Eğer operant2 içeriği operant1 içerisinde ise doğru olarak sonuçlanır.
NS	Karakter içermez	CS işleminin tersi durumdur.
CP	Örüntü içerir	Eğer örüntü karakter dizisinde varsa doğru olarak sonuçlanır.
CN	Örüntü içermez	CP işleminin tersidir.
AND	Ve	Eğer iki operatör doğruysa doğru olarak sonuçlanır.
OR	Ya da	Eğer herhangi bir operatör doğru ise doğru olarak sonuçlanır.
//	Yorum satırı	Yorun satırı başlangıcı için kullanılır.
/	Kaçış karakteri	Özel operantlar için karakter kaçış karakteri olarak kullanılır.
(	Sol parantez	Bir koşul ya da atama ifadesi başlatır.
)	Sağ parantez	Bir koşul ya da atama ifadesi bitirir.
;	Noktalı virgül	Satır sonu karakteridir.
IF, ELSEIF	Koşul başlangıçı	Koşul başlatma ön ekidir.
ELSE	Son koşul başlangıcı	Son koşul ifadesini başlatır.
	Boşluk	Operantlar ve operatör arasındaki ayraç olarak tanımlanmıştır.

Bu çalışmayla geliştirilen kural motoru iş akış motoruyla bütünleşik çalışarak ihtiyaç duyulduğunda çalışma zamanlı olarak takip edilen objeye müdahale edebilmesine olanak sağlamaktadır.

Kural motoru içerisinde ifadeler ve ifadelere bağlantılı olarak çalışan atamalardan oluşmaktadır. Temel gösterimle bir ifadenin örnek bir işlemi nasıl çözdüğü de Şekil 8'de gösterilmektedir.

Bir ifade, bir operant ile birlikte operantın sağında ve solunda operantların bulunduğu semantik bir yapıyı ifade etmektedir.

Bir operant sağında ve solunda bulunan iki operatöre bağlı olarak çeşitli karşılaştırma operasyonları uygulayarak ifadenin doğru veya yanlış sonuçlanması için kullanılmıştır. Bir koşul ifadesinin sonucu yalnızca doğru veya yanlış olarak sonuçlanabilir. İfadenin sonucunun doğru olduğu durumda ifadeyi takip eden atama işlemi gerçekleştirilir.

Operatörler sistem içirişinde önceden tanımlanmış olan değişkenler ya da iç içe çözülmüş olan sorgunun sonucu olan mantıksal doğru ya da yanlış ifadeleri gösterirler. Operant olarak ise anlamları Tablo 2'de verilen sabitler tanımlanmıştır.

Belge takip sistemi üzerinde kural motorunun örnek bir uygulaması için aşağıdaki formül

ifadesi örnek olarak Şekil 8'de verilmiştir. Burada "*current_state*", "*user_group*" ve "*doc_type*" program içerisinde önceden tanımlanmış değişkenleri ifade eder. Tek tırnak (') içerisinde gösterilen değerler ifadeler içerisinde kullanılan sabitlerdir.

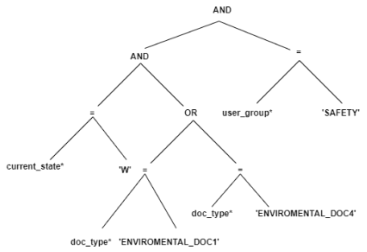
```
IF ( (
. . ( current_state EQ 'W' ) AND
. .( ( doc_type EQ 'ENVIROMENTAL_DOC1 ) OR
. . . ( doc_type EQ 'ENVIROMENTAL_DOC4 )
. . )
) AND
( user_group EQ 'SAFETY' )
{
// atama işlemi
}
```

Şekil 8. Basit birleştirme modeli

Sisteme girdi olarak verilen yukarıdaki örnek formül sistem içerisinde sanal bir ağaç yapısına dönüştürülür. Girdideki parantezlerin derinlik seviyelerinin ağacın derinliğini belirlediği bu sistemde yukarıdaki formülün ağaç yapısındaki gösterimi Şekil 9'da gösterilmektedir.

Kuralların işlenişi için kullanılan algoritmanın sözde kodları Algoritma 1'de verilmiştir. "kural_isle" fonksiyonu içerisinde girdi ilk olarak anahtar kelimelere göre koşullara bölünür. Her koşul ilk karakter kümesinin bölümlenmesinin ardından, Algoritma 2'de açıklanan "ifade_isle" fonksiyonuna girdi olarak verilerek işlenir.

"ifade_isle" fonksiyonu koşul ve koşula bağlı atama işlemini girdi olarak alır ve koşul ile ilgili karakter kümesini "ifade_cöz" fonksiyonuna gönderilir. Eğer koşul doğru ise atama işlemi için atama ile ilgili bölüm "atama_isleme" fonksiyonuna gönderilir.



Şekil 9. Örnek ağaç yapısı

Fonksiyon kural_isle (String input)

```
1. Input'u "IF" "ELSEIF" "ELSE"
keywordlerine göre böl
2. Bölünmüş ögeleri ifade dizisine
ata
3. Loop ifade[] into _ifade
4.     fonksiyon_cagir:
ifade_isle( _ifade, is_break )
5.    IF is_break = true
6.       return
7.    ENDIF.
8. Endloop.
```

Algoritma 1. Kural işleme

Fonksion ifade_isle (String ifade, boolean is_break)

```
1._ifadeyi '{' karakteristiğine
göre ikiye böl; _ifade , _atama
2. is_break =  fonksiyon_cagir
ifade_coz( _ifade );
3. IF is_break = true;
4.  fonksiyon_cagir atama_islemi (
_atama )
```

Algoritma 2. İfade işleme

Algoritma 4'de açıklanan "ifade_coz" fonksiyonu girilen koşul ifadesinin doğru ya da yanlış olarak sonuçlandığı fonksiyondur. Özyinelemeli (recursive) formatta çalışan fonksiyon her bir tekrarda girdi olarak verilen ifadenin çözülmesi ile küçülerek devam eder. Fonksiyondan çıkış koşulu 1. ve 3. satırlar arasında gösterildiği gibi girdinin doğru ya da yanlış olarak belirlenebildiği durumda tetiklenir.

Fonksiyon kendi içerisinde iç içe girmiş koşul ifadelerinden en iç seviyedeki parantezleri belirleyerek işleme başlar. 5. ve 14. satırlar arasında koşul ifadeleri atamik ifadeleri bulmak için tasarlanmıştır. Ağaç yapısının yaprak bölümlerinde en iç seviyedeki koşul ifadelerinin operantları yerleştirilir.

Açık parantezden sonra gelen karakter dizisinde bir sonraki parantezin açık parantez olması durumunda atomik olarak en derin ifadenin bulunduğunu gösterir. Döngü içerisinde girdi ifadesinin karakterleri dolaşılarak satır 20 ve 23 arasında atomik ifadeler bulunur ve ilgili indeksler saklanır. Tek bir döngü içerisinde tüm atomik ifadelerin indeksleri belirlendiğinden, bu döngü içerisinde bulunan indeksler arasındaki karakter dizisi Algoritma 3'de verilen "ifade_dogrula" fonksiyonuna gönderilir. Bu fonksiyon Tablo 2'de verilen operatörler ile ifadeyi analiz ederek doğru ya da yanlış olarak geri döner.

Fonksiyondan dönen doğru ya da yanlış karakteri, temel girdideki belirlenen indeksler arasındaki karakter kümesi ile değiştirilir. Bu adım sonrasında oluşturulan sanal ağaç yapısının en alt seviyesindeki ifadeler çözülmüş olur.

```
Fonksiyon ifade_dogrula returns string ( ifade ).
1.İlk ve son karakteri sil
2.Split into operant1 operator
operant2
3.operant1, operant2 değerlerini
getir
4.Tablo 2'deki mantığı uygula
5.Return result (doğru ya da
yanlış)
```

Algoritma 3. İfade doğrulama

```
Fonksiyon ifade_coz return bool ( _ifade )
1. IF _ifade = true OR _ifade =
false;
2.RETURN _ifade;
3. ENDIF.
4. LOOP _ifade i++.
5.  IF _ifade[i] EQ '('.
6. check if previous character is
escape symbol
7. IF prev. is escape symbol
8.  continue loop;
9. ENDIF.
10. LOOP _ifade j++
11.      IF j > i.
12.        IF _ifade[j] EQ '('
13.           check if previous
chracter is escape symbol
14.           IF _ifade[j-1] NE
escape_char.
15.        RETURN;
16.   ENDIF.
17.       ELSEIF _ifade[j] EQ ')';
18.           check if previous
chracter is escape
symbol
19.    IF _ifade[j-1] NE
escape_char;
20.                index-start =
i;
21.                index-end   =
j;
22.                index-ifade =
_ifade[i-j];
23.              APPEND index;
24.          ENDIF.
25.   ENDLOOP.
26. ENDLOOP.
27.   LOOP index k++.
28.      return = fonksiyon_cagir
ifade_dogrula( _index-ifade ).
29.   _ifade[index[k]-start,
index[k]-end] = return;
30.   ENDLOOP.
31. fonksiyon_cagir: ifade_coz
(_ifade )
```

Algoritma 4. İfade çözme

3. Bulgular

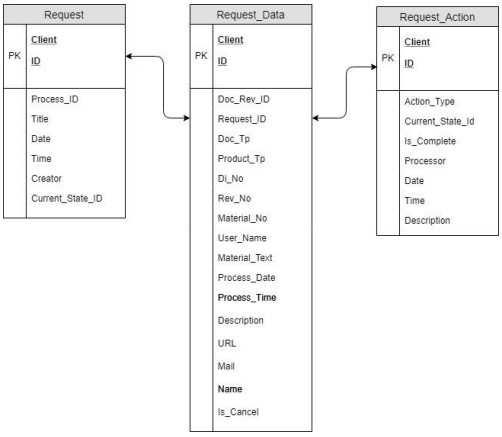
Çalışma uygulama olarak tamamlandıktan sorna bir firmada gerçek veriler üzerinde denenmiştir. Bu sayede firmadaki manuel, fiziksel olarak kâğıt veya e-posta yoluyla işleyen süreç; yazılım içerikli bir sisteme dönmüştür. Bu süreçte akışların dinamik olarak yaratılabiliyor olduğu görünsede akış yapısının ve bağımlılıkların sistem performansını etkileyebileceği fark edilmiştir. Bu nedenle tekrar edilen işlem maliyetinden kurtulmak adına, belge talep işleminin gerçekleştiği adımda toplanan veriler üzerinden bir sıralı örüntü madenciliği çalışması yürütülmüştür.

Bu çalışmada Apriori yöntemine benzeyen, genelleştirilmiş örüntü madenciliği algoritması (GSP) kullanılmıştır. Genel işleyişinde GSP algoritması Apriori algoritmasından çok farklı olmasa da; sadece bir öğe için değil de dizi sırası

için işlem yapılmasını gerçekleştirmesi bakımından birbirinden ayrılmaktadır [33].

Bu nedenle veri hazırlama süreci farklılık göstermektedir. Ham veriden işlenmeye hazır olan verinin sağlanması sürecinde; genel ilişkisel veri analizi sürecinde geçersiz verinin kullanımı açısından genel olarak iki farklı yöntem bulunmaktadır. Bu süreçte geçersiz veri analiz sürecine hiç katılmaz, görmezden gelinir ya da geçersiz veri, geçerli verinin ortalaması olarak kullanılır. Ancak sıralı örüntü madenciliğinde, sıralı frekans örüntüleri incelendiğinden bu iki yöntem de tercih edilmemiş ve gereksiz verinin tamamı anlamlı tek tip olacak şekilde varsayılmıştır [34].

Bu çalışmada veri madenciliği aracı olarak WEKA [35] kullanılmış ve veri özelleştirilmiş üç tablo üzerinden bir programla veriler .txt formatında dışarıya aktarılmıştır. Kullanılan tabloların yapısı Şekil 10'da verilmiştir.



Şekil 10. Veri kümesi oluşturulurken kullanılan tablolar

Veri kümesini oluşturabilmek için canlı sistemden alınan tablolar lokal alana txt formatında indirilmiş ve sonrasında dosya işlemleri yapabilen bir program üzerinde WEKA aracına girdi olarak kullanabilecek bir formata dönüştürülmüştür. Program içerisinde aşağıda detayları verilen tablolardaki verilerden girdi olarak kullanılmıştır.

Request: İş akışı motorunda isteği yapılan nesneye ait genel bilgiyi saklar.

Request_Data: İş akışı motorunun her bir farklı amaç için gerçekleştiriminde farklılaşan istek tablosudur. Belge takibi için kullanılan formda içerisinde isteği yapılan malzeme için ek alanlar bulundurur.

Requset_Action: İsteği yapılan obje üzerinde bir *t* anına kadar gerçekleşen aksiyonları saklar. Bu tablo bir log yapısı olarak da değerlendirilmektedir.

Sıralı örüntülerin arandığı yapılarda uzun metin ifadeleri performansı olumsuz yönde etkilediğinden ilk olarak belge türleri için etiketler oluşturulmuştur. Bu noktada geçerli olmayan, gerçek olmayan bazı veriler olduğu görülmüştür. Sistemin kurulumu ya da ileriki zamanlarda son kullanıcı kaynaklı bazı hatalar sonucu bu gerçek dışı görünen durumların oluştuğu/ortaya çıktığı değerlendirilmiştir. Ancak alışılagelmiş yöntem olan bu verilerin yok sayılması bu çalışma için uygun bulunmamıştır. Çünkü silinen bir belge talebi, bir sıralı ilişkiyi bozacağı ve belki de mümkün olan en uzun sıralı örüntünün kaybolmasına sebep olacağı için değerlendirme dışına alınmamıştır. Bu süreçte geçersiz verinin işlenmesi için farklı bir yöntem kullanılmıştır. Geçerli olmayan sıralar için yeni bir etiket kullanılarak tüm veriyi eksiltmeden işlem yapılabilecek bir veri kümesi oluşturulmuştur.

Girdi olarak dosya alan program, çıktı olarak da WEKA aracına girdi olabilecek formatta içerisinde zamanında bulunduğu bir dosya ortaya çıkarmaktaktadır. Veri kümesi sadece belge etiketleri, tarih ve saat içerecek şekilde oluşturulmuştur.

Tarih ve saat birleştirilerek ayrıca sıralanmış ve tekrarsız bir sayı dizisine dönüştürülmüştür. Son durumda veri seti içerisinde yalnızca sırayı gösteren sayılar ve belge isimlerini betimleyen etiketler bulunmaktadır. Örnek bir veri kümesi Şekil 11'de gösterilmiştir.

```
@relation sequential_test_set

@attribute day {1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 2
@attribute 'doc label' {doc=A, doc=B, doc=C, doc=D, doc=E, doc=F,
@attribute 'doc2 label' {doc2=A, doc2=B, doc2=C, doc2=D, doc2=E,
@attribute 'doc3 label' {doc3=A, doc3=B, doc3=C, doc3=D, doc3=E,
@attribute 'doc4 label' {doc4=A, doc4=B, doc4=C, doc4=D, doc4=E,
@attribute 'doc5 label' {doc5=A, doc5=B, doc5=C, doc5=D, doc5=E,
@attribute 'doc6 label' {doc6=A, doc6=B, doc6=C, doc6=D, doc6=E,
@attribute 'doc7 label' {doc7=A, doc7=B, doc7=C, doc7=D, doc7=E,

@data
1,doc=A,doc2=A,doc3=A,doc4=A,doc5=A,doc6=A,doc7=A
2,doc=B,doc2=A,doc3=A,doc4=A,doc5=A,doc6=A,doc7=A
3,doc=C,doc2=A,doc3=A,doc4=A,doc5=A,doc6=A,doc7=A
4,doc=A,doc2=B,doc3=A,doc4=C,doc5=A,doc6=A,doc7=A
5,doc=A,doc2=A,doc3=A,doc4=A,doc5=A,doc6=A,doc7=A
```

Şekil 11. Örnek bir veri kümesi

En düşük destek değerinin 0,6 belirlendiği test sonuçlarında iki adet belge etiketi çifti, en düşük destek değerini 0,9 belirlendiği test sonuçlarında da bir adet belge etiketi çifti sınırın üzerinde hesaplanmıştır.

4.Sonuç

Bu çalışma kapsamında aktivite tabanlı bir kural motoru geliştirilmiştir. Geliştirilen bu yapı, iş akış motorunun en temel özelliklerinden tekrarlı yapıları sağlayabilmesi için bir kural motoru ile desteklenmiştir.

Fabrikanın manuel olarak yönettiği bu süreç otomatize edildiğinden raporlamaya uygun hale getirilmiş ve belge talepleri arasında sistemde bir kuyruk yapısının oluşması sağlanmıştır. Akışın sistem üzerinden yürümesi belgelerin yalnızca ilgili kişilere ulaşmasını sağladığından sisteme bir güvenlik önlemi katkısı da sunmuştur.

Ayrıca projenin canlı sisteme geçişinden sonra altı ay veri girişi takip edilmiş ve işlemsel log dosyaları sıralı örüntü madenciliği uygulamasının giriş dosyası olacak şekilde formatlanmıştır. Uygulama sonuçları ilgili iş birimleri ile paylaşılmış ve sistemin daha hızlı işlemesi için bu araştırma sonuçları kullanılmıştır.

Kaynakça

[1] Chung S., Synder C. 1999. ERP initiation -a historical perspective, Proceedings of AMCIS'99, s. 213-215.

[2] Ouyang C., Adams M., Wynn M., Ter A. 2010. Workflow Management, Handbook on Business Process Management 1: Introduction, Methods, and Information Systems, s. 387-418 DOI: 10.1007/978-3-642-00416-2.

[3] Marek R, Amit S. 1995. Specification and execution of transactional workflows. In Modern database systems, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, s. 592-620.

[4] Bergmann S. 2007. Design and implementation of a workflow engine", Diploma thesis of, Rhenish Friedrich Wilhelm University, Siegburg 92s.

[5] McCoy, D. W. and Sinur J., Achieving Agility: The Agile Power of Business Rules, Gartner, Special report on Driving Enterprise Agility, 20 April 2006.

[6] De Leusse P., Kwolek B., Zielinski K. 2012. A common interface for multi-rule-engine distributed systems, Distributed System Research Group, AGH University of Science and Technology Krakow, Poland

[7] Diimitrios G., Mark H., Amit S. 1995. A n overview of workflow management: from process modeling to workflow automation infrastructure, Distributed and Parallel Databases Cilt 3, s 119-153.

[8] Wil van der Aalst, Kees van Hee 2000. "Workflow management models, methods and systems", Eindhoven, Hollanda.

[9] Bouafia, K., Molnár B. 2018. Dynamic business process: comparative models and workflow patterns, The 11th conference of phd students ın computer scıence volume of short papers CS2. 194, June 2018 Szeged, Magyarország.

[10] Krasimira P. S. Todor A. S. 1999. Evolution of the workflow management systems, Information system, Cilt 28 s 27-60.

[11] De Lay, E, Jacobs D. 2011. Rules-based analysis with JBoss Drools: adding intelligence to automation, Proceedings of ICALEPCS 2011, s. 790-793.

[12] Rakesh A., Dimitrios G., Frank L. 1998. Mining Process Models from Workflow Logs In Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '98), Hans-Jörg Schek, Fèlix Saltor, Isidro Ramos, and Gustavo Alonso (Eds.). Springer-Verlag, Berlin, Heidelberg, s 469-483.

[13] Aalst W.M.P., F. van Dongen B., Herbst J. Măruşter L., Schimm G., Weijters A. 2003. Workflow mining: a survey of issues and approaches. Data & Knowledge Engineering. Cilt 47(2) s 237-267 DOI: 10.1016/S0169-023X(03)00066-1.

[14] Singh G. N., Sandeep A. 2011. A process model for workflow mining, International Journal of Information Technology and Knowledge Management, Cilt. 4(2) s. 719-722.

[15] Allen, R. 2001 Workflow: An Introduction" in: WfMC Workflow Handbook 2001, L. Fischer (ed.), Future Strategies Inc, Florida US, s15-38.

[16] Lucas, S. M., Reynolds, T. J. 2005. Learning deterministic finite automata with a smart state labeling evolutionary algorithm. IEEE Transactions on Pattern Analysis & Machine Intelligence, Cilt 7, s. 1063-1074. DOI: 10.1109/TPAMI.2005.143

[17] Xiangru XU, Yiguang H. 2012 .Matrix expression and reachability analysis of finite automata. J Control Theory Appl, 10(2) s 210-215.

[18] Molnár, B., Máriás Z. 2015. Design and Implementation of a Workflow Oriented ERP System, International Conference on e-Business, s. 160-167.

[19] ZETA Workflow http://zetacomponents.org /news.html (Erişim Tarihi 10.06.2019)

[20] Van Der Aalst, W. M., Ter Hofstede, A. H. 2005. YAWL: yet another workflow language. Information systems, Cilt. 30(4), s. 245-275. DOI: 10.1016/j.is.2004.02.002

[21] Pirt, B. 2004. Learning eZ publish 3: Building Content Management Solutions. Packt Publishing Ltd.

[22] Galaxia Workflow https://galaxyproject.org/ learn/advanced-workflow/ (Erişim Tarihi 10.06.2019)

[23] Radicore Workflow. https://www.radicore.org/ viewarticle.php?article_id=3 (Erişim Tarihi 10.06.2019)

[24] Visual Workflow https://www.adobe.com/tr/ marketing/experience-manager-forms/visual-workflow-editor.html (Erişim Tarihi 10.06.2019)

[25] Verve Worklow http://www.lateralsystems.com.au/verve-cms-1.html (Erişim Tarihi 10.06.2019)
[26] Staffware http://www.workflowpatterns.com/vendors/staffware.php (Erişim Tarihi 10.06.2019)
[27] MQSeries Workflow https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/fg15350_.htm (Erişim Tarihi 10.06.2019)
[28] Ran, S., Brebner, P., Gorton, I. 2001. The rigorous evaluation of Enterprise Java Bean technology. In Proceedings 15th International Conference on Information Networking. s. 93-100. DOI: 10.1109/ICOIN.2001.905336
[29] HP ChangeEngine http://www.hp.com/hpinfo/newsroom/press_kits/2009/HPSoftwareUniverseHamburg09/HPOODataSheet.pdf (Erişim Tarihi 10.06.2019)
[30] Fujitsu i-Flow http://www.iflowbpm.com/ (Erişim Tarihi 10.06.2019)
[31] SAP R/3 Workflow https://wiki.scn.sap.com/wiki/pages/viewpage.action?pageId=473964457 (Erişim Tarihi 10.06.2019)
[32] Swamynathan, S., Geetha, T.V. 2014 Active rule engine for dynamic business rules https://www.researchgate.net/publication/249903201_ACTIVE_RULE_ENGINE_FOR_DYNAMIC_BUSINESS_RULES (Erişim Tarihi 10.06.2019)
[33] Desai, N., Ganatra, A. 2014. Draw Attention to Potential Customer with the Help of Subjective Measures in Sequential Pattern Mining (SPM) Approach, Conference: Proc. of Int. Conf. on Recent Trends in Information, Telecommunication and Computing, ITC.
[34] Zhang S., Zhang C., Yang Q. 2003 Data preparation for data mining. Applied Artificial Intelligence,Cilt 17(5-6) s. 375-381.
[35] WEKA, Open source machine learning tool. https://www.cs.waikato.ac.nz/ml/weka/ (Erişim Tarihi 10.06.2019)