BİLGİSAYAR BİLİMLERİ VE TEKNOLOJİLERİ DERGİSİ
JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGIES
http://dergipark.org.tr/tr/pub/bibted

Araştırma Makalesi

Yazılım Risklerinin Doğasına Uygun Yöntem: Bulanık Mantık

Mustafa Batar*[1], Kökten Ulaş Birant[2], Ali Hakan Işık[3]

[1]*Dokuz Eylül Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, İzmir, Türkiye*
[2]*Dokuz Eylül Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, İzmir, Türkiye*
[3]*Burdur Mehmet Akif Ersoy Üniversitesi, Mühendislik-Mimarlık Fakültesi, Bilgisayar Mühendisliği Bölümü, Burdur, Türkiye*

Anahtar Kelimeler:
Yazılım Riskleri
Belirsizlik
Yapay Zekâ
Bulanık Mantık
Bulanık Yaklaşım

ÖZ

"Bulanık Mantık", insanlara "Klasik Mantık" da olduğu gibi ikili cevaplar (doğru/yanlış, evet/hayır, iyi/kötü, siyah/beyaz, vb.) vermek yerine belirsizliği de bir seçenek olarak ele alarak derecelendirme, yüzdeliklere ayırma, olasılıklara dayalı ya da oran orantı doğrultusunda cevaplar verebilmektedir. Bilgisayarın çalışma mantığının aksine, 1 ya da 0 veya doğru ya da yanlış demek yerine; çok doğru, az doğru, belirsiz, az yanlış, çok yanlış seçeneklerini bizlere cevap olarak sunabilmektedir. Günümüzde hemen hemen her alanda etkin bir şekilde kullanılan "Yapay Zekâ"nın içerisinde yer alan "Bulanık Mantık", insan düşünce yapısına uygun bir şekilde hareket ederek, bir olayla ilgili karar verme mekanizmasını rahatça çalıştırabilmektedir. İnsan doğası gereği, karar verme yapısında net olmayan durumlar her zaman olabilmektedir. Kesin olmayan bu durumlar doğrultusunda, "belirsizliği" de içerisinde barındıran yazılım risklerinin belirlenmesi ve tanımlanması gerekliliği ortaya çıkmaktadır. Yazılım riskleri ele alınırken hem kendi yapısı ile örtüşen hem de proaktif yöntemler uygulanmalıdır. Bu sebepten dolayı, yazılım projelerinin başarısını düşürme gücüne sahip yazılım riskleri, kendi (çalışma) mekanizması ile uyumlu hareket edebilen bir yöntem ile etkin bir şekilde ortaya çıkartılıp bertaraf edilmelidir. Bu çalışmada, "Bulanık Mantık" yöntemine bağlı olarak "belirsizlik" kavramını bir seçenek olarak sunan ve içinde bulunduran "Bulanık Yaklaşım" tekniğiyle, yazılım geliştirme sürecinde negatif yönde bir etkiye sahip yazılım riskleri, doğasına uygun bir şekilde etkin olarak belirlenip tanımlandığı kendine has örnekleriyle detaylı bir şekilde anlatılmaya çalışılmıştır.

Appropriate Method for Software Risks' Nature: Fuzzy Logic

Keywords:
Software Risks
Uncertainty
Artificial Intelligence
Fuzzy Logic
Fuzzy Approach

ABSTRACT

"Fuzzy Logic" can give answers based on grading, separating percentages, probabilities or ratios by taking uncertainty as an option instead of giving people binary answers (true or false, yes or no, good or bad, black or white, etc.), as in "Classical Logic". In addition, instead of giving answer as one or zero, right or wrong as in the computer working principle, it is able to present more right, less right, uncertain, less wrong or more wrong options as an answer to us. "Fuzzy Logic" included in the "Artificial Intelligence", which is used and applied in almost all fields nowadays, can easily operate the decision-making mechanism regarding an event by acting in accordance with the human mind. Due to human nature, there can always be situations that are not clear in the decision-making structure. In line with these uncertain situations, the necessity of identifying and defining the software risks that contain "uncertainty" emerges. Thus, both structure overlapping and proactive methods has to be applied when addressing these software risks. For this reason, software risks that have the power to reduce the success of software projects has to be identified and eliminated effectively with a method that is able to act in harmony with software risks' own mechanism. In this study, with the contribution of "Fuzzy Approach" technique, which presents and includes the concept of "uncertainty" based on the "Fuzzy Logic" method, it has been tried to be explained in detail with specific examples how software risks that have a negative effect in the software development process are determined and defined effectively in accordance with their nature.

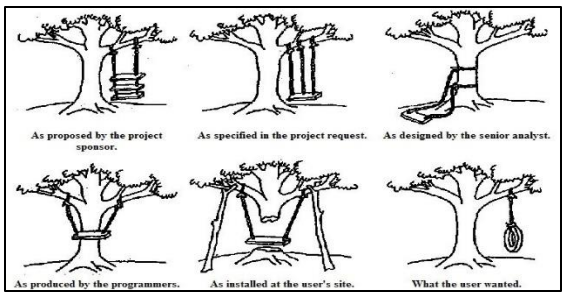
***Sorumlu Yazar**

*(mbatar@cs.deu.edu.tr) ORCID ID 0000-0002-8231-6628
(ulas@cs.deu.edu.tr) ORCID ID 0000-0002-5107-6406
(ahakan@mehmetakif.edu.tr) ORCID ID 0000-0003-3561-9375

e-ISSN: 2717-8579
Geliş Tarihi: 22/03/2021; Kabul Tarihi:06/04/2021

1. GİRİŞ

Yazılım geliştirme projelerinde, oldukça büyük bir bütçe ve finansal planlamaya ihtiyaç duyulmaktadır ve geliştirmeye devam edilebilmesi için çok büyük yatırımlar gerekmektedir. Yazılım geliştirmeyle ilgili uluslararası hacimde önemli bilgilere bağlı maliyetlere bakıldığı zaman; 1985 yılında 150 milyar dolar, 2010 yılında 2 trilyon dolar, 2015 yılında 5 trilyon dolar ve 2020 yılında ise 7 trilyon doların üzerindedir. Ayrıca, 2021 yılının ilk yeni günlerinde, Apple Store'un sadece bir günlük cirosu yaklaşık 500 milyon dolar civarındadır ve her geçen gün katlanarak bu meblağ artmaktadır. Bununla birlikte, her geçen yıl önemli ölçüde genişleyen ve artan harcamalara ve yatırımlara rağmen, yazılım geliştirme projelerinin başarısı hiç de yüksek değildir. 2015 yılında düzenlenen "CHAOS" raporuna göre, yazılım projelerinin sadece %17'si tahsis edilen finansal planda ve ihtiyaçlar doğrultusunda başarılı bir şekilde tamamlanmıştır. Fakat yazılım projelerinin %53'ü ise, istenilen süreyi aşarak veya potansiyel olarak harcama planının üzerine çıkarak veya ön koşulları tam olarak karşılayamadan eksik bir şekilde tamamlanmıştır. Ayrıca %30 oranında yazılım geliştirme projeleri tamamlanamayıp yarıda bırakılmıştır. 2020 yılında yayınlanan "CHAOS" raporunda ise, tüm dünyadaki yazılım projelerinin sadece %33'ünün başarıyla tamamlanabildiği kayıtlara geçmiştir. Vasatın altındaki bu oranla ilgili, yazılım projelerinin başarı ve başarısızlık durumunu tasvir etmeye çalışan görsel aşağıda Şekil 1'de verilmiştir. Bununla birlikte, bu denli başarısız sonuçlarla baş edebilmek, üstesinden gelebilmek ve yazılım geliştirme sürecinde başarı oranını artırabilmek için, yazılım risklerinin ortaya çıkarılmasını sağlayan etkin bir yöntemin kullanılması ve uygulanması gerekmektedir (Fairley, 1994). Bu yöntem doğrultusunda, yazılım geliştirme projelerinin başarılı bir şekilde tamamlanmasını engelleme gücüne sahip yazılım riskleri bir sorun yaratmadan önce, zamanında fark edilip ayırt edilebilecektir. Bu çalışmada, yazılım risklerini etkin bir şekilde tanıyan ve tanıtan yapay zekâ entegresi olan bulanık mantığa dayalı bulanık yaklaşım tekniği, vaka analizi çalışması ile beraber detaylı bir şekilde örnekleriyle ortaya konmuştur.



Şekil 1. Yazılım projelerinin başarısı/başarısızlığı (Fu vd., 2012)

2. RİSK

Risk, hedeflenen bir sonuca ulaşamama olasılığıdır. Ayrıca, bir organizasyonun ya da bir firmanın veyahut bir kişinin stratejik, finansal ve operasyonel hedeflerine ulaşmasını engelleyecek olay ya da olayların ortaya çıkma durumudur. Riskin temel olarak iki ana faktörü vardır: Gerçekleşme olasılığı; belirli bir sonuca ulaşamama olasılığı veya istenmeyen bir durumun meydana gelme olasılığıdır. Zararın boyutu, risklerin gerçekleşmesi durumunda ortaya çıkabilecek sonuçların etkileridir (Smith, 1989).

Mevcut alan-yazında üç ana risk kategorisi göze çarpmaktadır (Renn, 2004). İlk olarak, şirketle ilgili iç riskler ortaya çıkmaktadır: Üretim yönetiminin etkinliğine bağlı riskler, finansal yönetim faaliyetine ilişkin riskler, pazarlama yönetiminin etkinliğine bağlı riskler, şirket içi lojistik ile ilgili riskler, kalite yönetiminin etkinliği ile ilgili riskler, insan kaynakları yönetimine dayalı riskler, yönetimle ilgili genel riskler, vb. risklerdir.

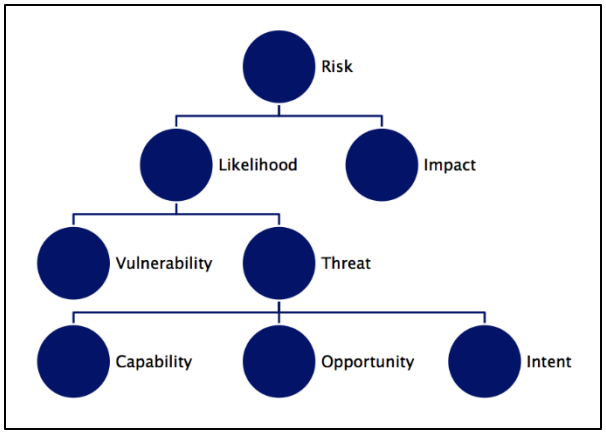
İkinci olarak, tedarik zinciri ağı ile ilgili riskler ortaya çıkmaktadır: Tedarikçilerin üretim girdisini istenilen miktarda tedarik edememe riski, tedarikçilere zamanında teslim edememe riski, tedarikçilerin istenilen kalite standartlarına ulaşamama riski, tedarikçi ve dağıtım şirketlerinin riskleri, satıcıların ve dağıtım şirketlerinin maliyetini şirket için kritik reçeteler ile karşılayamaması, şirketin hızlı teknolojik gelişmelere ayak uyduramaması, şirket için kritik reçeteye sahip tedarikçiler ve dağıtım şirketleri ile stratejik olmayan işbirliği riski, dağıtım şirketlerinin ürünlere zamanında ulaşamaması, ürünlerde hasar riski, dağıtım sırasında ürün kalitesinin düşmesi riski, stratejik öncelikleri başta olmak üzere tedarikçiler ve dağıtım şirketleri ile etkin bir bilgi ağının kurulamamasından kaynaklanan riskler, bilgi ağının etkin bir şekilde kullanılamaması, ortaya çıkabilecek riskler ile dış kaynak kullanımı yoluyla şirketin temel lojistik işlevlerini kısmen veya tamamen yerine getirememesi, vb. risklerdir.

Son olarak, dış riskler ortaya çıkmaktadır: Ekonomik belirsizliklerden kaynaklanan riskler, siyasi istikrarsızlık, teknolojik gelişmelerden kaynaklanan riskler, değişen yasal koşulların riskleri, sosyo-ekonomik durumdaki değişikliklerin yarattığı riskler, artan ve değişen rekabet koşullarının yarattığı riskler, müşteri beklentilerindeki büyük değişiklik, doğal afet riski, terör riski, vb. risklerdir.

Riskler insan hayatında tüm boyut ve şekillerde ortaya çıkabilmektedir. İşin uzmanları genel olarak ve çoğunlukla üç önemli noktayı ele almaktadırlar (Lezzoni, 1997): İlk olarak ele alınan piyasa riski; maliyetlerin bir organizasyonun ters sonuçlarına sahip olacak şekilde hareket etmesi tehlikesidir. İkinci olarak ele alınan kredi riski; bir müşterinin, karşı tarafın veya bir sağlayıcının taahhütlerini

yerine getirmeyi ihmal etmesi tehlikesidir. Üçüncü olarak ele alınan operasyonel risk; bireylerin, döngülerin veya çerçevelerin yetersiz kalması ya da dış bir durumun (sismik titreme, yangın, vb. durumlar) organizasyonu tersine – negatif yönde – etkilemesi tehlikesidir.

Ortaya konan riskler çerçevesinde, alanyazında belirlenen risklerin etkinlik düzeyi ile ilgili tanım kümesi bütün kademe ve parametreleri içerecek şekilde aşağıda Şekil 2'de gösterilmeye çalışılmıştır.



Şekil 2. Risk tanım kümesi (Hall, 1998)

3. YAZILIM RİSKLERİ

Yazılım geliştirme ve bu gelişimi sürdürme oldukça emek isteyen ve içerisinde risk barındıran tehlikeli bir iştir. Yazılım geliştirme süreci gereksinimlerin belirlenmesinden bakım, onarım ve güncelleştirmeye kadar devam etmektedir. Bununla birlikte, işin özünü oluşturan tasarım ve gerçekleştirme aşamalarıdır. Buna bağlı olarak, ek maliyetler, süreç bakımından gecikmeler veya çıktıları karşılamadaki etkisizlik gerçek sonuçlar doğurabilmekte ve başarısızlıklar ortaya çıkabilmektedir (Dey vd., 2007). Yazılım geliştirme sürecinde sıkıntılara yol açabilecek, süreci baltalayabilecek ve başarısızlığa neden olabilecek öne çıkan – ilk 10 – yazılım riskleri aşağıda listelenmiştir (Arnuphaptrairong, 2011):

Tahmin, Süreç ve Planlama: Bireysel yazılım projelerinin benzersiz oluşu; analizciler, tasarımcılar, geliştiriciler, testçiler ve yöneticiler için geliştirme süresini tahmin etme ve programlamada doğası gereği sorunlar yaratabilmektedir. Gelecekte öğrenilen dersleri uygulamak için daima mevcut yazılım projelerini incelemek ve takip etmek gerekir.

Gereksinimlerde Hızlı Büyüme ve Gelişme: Yazılım geliştirme sürecinde proje ilerledikçe, daha önce tanımlanmayan sorunlar, ihtiyaçlar ya da istekler, son teslim tarihine başarıyla yetiştirebilmek için son dakikada bir engel oluşturabilmektedir. Bu durumu önleyebilmek için, projenin başlarında büyük düşünmeye çalışmak ve en kötü durum ya da en ağır kullanım senaryosunu öngörmek gerekmektedir.

Çalışanların Sirkülasyonu, Değişikliği ya da Yer Değiştirmesi: Şirketin her bir yazılım projesinde çalışan birkaç geliştiricisi vardır. Bir geliştirici projeden ayrıldığında, kritik bilgileri yanına alabilmektedir. Bu durum, tüm projeyi geciktirebilir ve bazen de raydan çıkarabilir. Bu riskli durumu engelleyebilmek adına, proje takım üyelerinin işbirliği yapabileceği ve bilgileri paylaşabileceği kaynaklara sahip olmak gerekmektedir.

Spesifikasyon Dökümanındaki Eksiklik: Entegrasyon ve kodlamanın ilk aşamalarında yazılım projelerinde gereksinimler çatışabilmektedir ve kafa karışıklığına yol açabilmektedir. Bununla birlikte, geliştiriciler spesifikasyonun belirsiz, eksik ya da hiç olmadığını fark edebilirler. Bu olumsuz durumun önüne geçebilmek adına, spesifikasyon dökümanı ayrıntılı bir şekilde yazılmak zorundadır.

Verimlilik ve Üretkenlik Sorunları: Uzun zaman çizelgeleri ve planlamaları içeren yazılım projelerinde; geliştiriciler ve diğer takım üyeleri işe başlamak için kolaylık (kolaya kaçma) eğilimindedirler. Bunun bir sonucu olarak, projeyi tamamlayabilmek için önemli ölçüde bazen zaman kaybedebilirler. Bu zaman kaybını engelleyebilmek için, gerçekçi bir program belirlemek ve buna bağlı kalmak gereklidir.

Tasarımlardan Ödün Vermek: Geliştiriciler, bir sonraki "gerçek" görevlere geçebilmek için tasarım sürecini genel olarak aceleye getirme eğilimindedirler. Fakat tasarım, yazılım geliştirmenin en kritik parçası olduğundan, bu acelelik, programlama süresine ek zaman olarak yansıyacaktır. Bu zaman kaybını ortadan kaldırabilmek adına, yazılım geliştirme sürecinde iç ve dış tasarıma yeterince önem verilmek zorundadır.

Altın Kaplama/Servis: Yazılım geliştirme sürecinde, proje takım üyeleri bazen gereksiz özellikler ekleyerek becerilerini sergilemeyi severler. Örneğin, bir geliştirici, "şık" görünmesi için temel bir giriş modülüne Flash mekanizmasını boş yere ekleyebilir. Bu da, programlama saatlerinin boşa harcanması anlamına gelmektedir. Gereksiz zaman kayıplarının önüne geçebilmek adına, temel fonksiyonel görevler ekleme yapmadan yerine getirilmeli ve gerçekleştirilmelidir.

Prosedürel Riskler: Yazılım geliştirme sürecinde, günlük operasyonel faaliyetler; hatalı süreç uygulaması, çakışan öncelikler veya proje takım üyelerinin sorumluluklarının net bir şekilde belirlenmemesi nedeniyle aksayabilir. Bu aksamanın ortadan kalkması için süreç boyunca istenilenlerle ilgili net ve açık olmak gerekmektedir.

Teknik Riskler: Yazılım geliştirme sürecinde, yazılım şirketleri, yüksek bütçeler ve zamanlamayla ilgili fazlalıkları telafi edebilmek için yazılımın işlevselliğini azaltabilmektedirler. Yazılımın maksimum işlevselliğine ulaşmak ile en yüksek performans arasında her zaman bir çelişki ortaya çıkmaktadır. Bu sebepten dolayı, yazılım firmaları, bütçe aşımını dengeleyebilmek ve zaman aşımlarını önleyebilmek adına bazen yazılımın fonksiyonelliğiyle oynayabilmektedirler. Bunun neticesi olarak da teknik sorunlar ortaya çıkabilmektedir. Bu sorunların ortaya çıkmaması

adına, bütçe, zaman ve planlamanın ortak bir akılla yapılması her daim gerekmektedir.

Öngörülemeyen/Kaçınılmaz Riskler: Bu tür riskler; hükümet politikasındaki değişiklikleri, yazılımın eskimesini, kontrol edilemeyen ya da tahmin edilemeyen diğer riskleri içermektedir. Yazılım geliştirme alanı ve süreci giderek daha karmaşık hâle gelmekte ve buna bağlı olarak öngörülemeyen riskler daha da artmaktadır. Yazılım firmalarının bu tür riskleri azaltabilmesi ve önüne geçebilmesi adına, stratejik planlamaya odaklanması hayati önem arz etmektedir.

Yazılım geliştirmede aktif bir şekilde ortaya çıkan bu 10 riskin süreç boyunca hangi temel alana ve aşamaya etki ettiği aşağıda Şekil 3'te ifade edilmeye çalışılmıştır.

Şekil 3. Yazılım riskleri (Hoermann vd., 2012)

4. BULANIK YAKLAŞIM

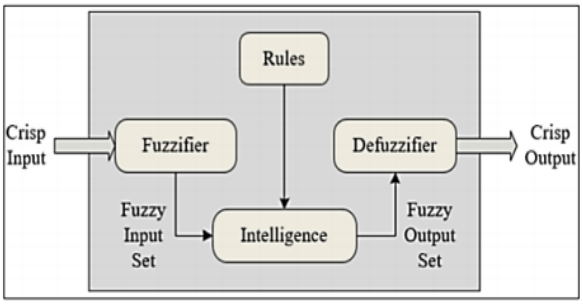
"Bulanık Mantık", Boolean yönteminde yer alan "geçerli ya da sahte" durumu yerine "kesinlik seviyelerine" dayalı bir yaklaşım olarak ele alınır. 1960'larda Dr. Lotfi Zadeh bulanık mantık yapısını Berkeley'deki California Üniversitesi'nde kendi derslerinde ilk olarak uygulamıştır. "Fluffy hipotezi", sezgisel verilerle doğrusal olmayan ilişkiler kurmada şüpheli ya da belirsiz bir yapı kurmak için bir yöntem olarak kullanılabilir. Bu hipotez esas olarak, bir eklemlenmenin 0 ya da 1 olmasından ziyade, değerinin bu aralıkta değişebilen bir seçeneğe veya cevaba sahip olabileceği mantığıyla çalışır (Ross, 2017).

"Bulanık Yaklaşım", her bir katkısının etkisinden bahsedebilmek için kullanılan grafik bir tasvirdir. Bilgi ve etki alanı, genel olarak, bir çerçevede meydana gelebilecek tüm koşulları iletebilen yaygın bir küme olarak karakterize edilir. "Fluffy mantığı", bilgi eklemlerini belirli bir ağırlığa göre standartlaştırır. Ayrıca bu noktada, bir verim, bir oran ya da bir yüzdelik değerinde ortaya çıkan çerçeveyi etkin bir şekilde görüntülemek için katkılar ya da katmanlar arasındaki bağlantıları karakterize eder. Tanımlanan kurallar, bilginin etkilerine karar vermek ve nihai ürünün bulanık mantıksal kümeleri üzerindeki artikülasyonları ortaya çıkarabilmek için ağırlıklandırma ya da derecelendirme bileşeni olarak karakterize edilir. Kapasitelerin oluşturulması, gözden geçirilmesi ve birleştirilmesi, çerçeve için etkili bir şekilde ortaya çıkan bulanık bir mantık getirisi üretir. Bulanık mantıklı tüm bilgi ve verim ifadeleri, farklı katılım ve içerik kapasitelerine sahiptir. Yönergeler ya da kurallar, "ALSO (Ayrıca)" veya "AND (ve)" sayısal ibareler kullanılarak gerçekleştirilen bir dizi karakterize karar içerir. Bulanık mantık, karakterize edilmiş girdi eklemlenmelerinden fonetik niteliklere ve bulanık kümelere dönüşür (Shen ve Chouchoulas, 2002).

"Fluffy mantığı" nın Takagi-Sugeno tekniği ve Mamdani tekniği olarak adlandırılan çeşitli stratejileri mevcuttur. Takagi ve Sugeno tarafından önerilen Tagaki-Sugeno tekniği, doğrusal olmayan çerçevelerde bulanık mantık bilgisini belirlemek ve artikülasyonlar elde etmek için kullanılan bulanık mantık stratejisidir. Takagi-Sugeno modelinde bilgi ve verim artikülasyonları "IF-THEN (Eğer… İse… O halde)" standartları ile karakterize edilir. İlkeler ve kurallarda belirtildiği gibi, verim ve çıktı temel bir prosedür yardımıyla belirlenebilir. Bununla birlikte, Mamdani bulanık mantık tekniğinde, katılım kapasitelerinin bulanıklaştırılmasına uyabilmek için çok sayıda kural ve kurallar bütünü karakterize edilmelidir (Hájek, 1998).

"Bulanık Yaklaşım", kişisel bilgisayarın genel çalışma mantığını, bireylerin mantık sistemi ve düşünce yapısı içinde kavrayabilecekleri şekilde modellemeye çalışır. Bir bilgisayarın mantık bloğu, müşteriden doğrudan katkı almakta ve "EVET" veya "HAYIR" sonuçlarına eşit olan "DOĞRU" veya "YANLIŞ" çıktılarını vermektedir. Bununla birlikte, bulanık yaklaşıma göre, müşterinin seçiminde "EVET" ve "HAYIR" arasında çeşitli olası sonuçların ve çıktıların olduğu ifade edilir. Bulanık mantık stratejisi kullanılarak, belirsiz durumların, uygun olmayan şekilde karakterize edilmiş veya karmaşık çerçevelerin gösterilmesi amaçlanmıştır (Carlsson ve Fuller, 2003).

"Bulanık Mantık"a dayalı "Bulanık Yaklaşım" mimarisi, aşağıda görüldüğü gibi üç temel bölümden ve alandan oluşmaktadır. Başlangıçta, bulanıklaştırma modülünde verilen bulanık setlere göre çerçeve katkılarını değiştirir. Standartlar ya da kurallar alanı, çerçevenin bulanık mantık yaklaşımının getirilerini belirleyen koşulları tasvir eder. Bu koşullar, çerçevenin değişen bilgi eklemlenmelerine karşı hangi eklemlenmenin sağlanması gerektiğini gösterir. Sonunda, bulanıklaştırma modülü, tahmin motoru tarafından üretilen bulanık sete göre net bir değere dönüşür. Bu süreç boyunca, çıktılar belirlenen kurallara bağlı olarak çeşitli değerler vermektedir (Inyang ve Joshua, 2013). Bununla birlikte, "Bulanık Mantık"a dayalı "Bulanık Yaklaşım" tekniğinin çalışma mantığı, süreci ve aşamaları aşağıda Şekil 4'te gösterilmeye çalışılmıştır.



Şekil 4. Bulanık yaklaşımın çalışma mantığı (Karataş vd., 2020)

5. BULANIK YAKLAŞIM YÖNTEMİYLE YAZILIM RİSKLERİNİN TANIMLANMASI

Bulanık yaklaşım yönteminde genel olarak ibareler dilsel ve mantıksal kurallar ile ifade edilir. Bu ortaya konan kurallar sayesinde yazılım riskleri belirlenip tanımlanabilir (Jiang ve Klein, 2000). Yazılım riskleri kendi içerisinde "belirsizliği" barındırdığı için, "belirsizliği" ana ögesi haline getiren bulanık yaklaşım tekniğiyle oldukça uyumlu hareket edebilmektedir. Bu uyum sayesinde, yazılım riskleri yazılım uzmanlarının görüşleri doğrultusunda bulanık mantığa dayalı bulanık yaklaşım yöntemi ile çıktıları ile beraber ortaya etkin bir şekilde konabilir (Keil vd., 1998). Yöntem bazlı geliştirilen yazılım risklerinden örnekler (13 adet Yazılım Riskine bağlı kurallar bütünü) aşağıda listelenmiştir (Birant vd., 2019).

1-Eğer "yazılım geliştirme sürecinin olgunluğu" yüksek ve "yazılım ekibinde çalışanların zaman dilimi farklılığı" düşük ise, o halde Yazılım Riski düşüktür.

2-Eğer "yazılım geliştirme sürecinin düzenli işleyiş seviyesi" yüksek ve "yazılım ekibinde çalışanların konuştukları dil farklılığı" düşük ve "yazılım geliştirme sürecinde çalışanlar arasındaki iletişim düzeyi" yüksek ise, o halde Yazılım Riski düşüktür.

3-Eğer "yazılım geliştirme sürecinde şeffaflık düzeyi" yüksek ve "yazılım ekibinde çalışanların zaman dilimi farklılığı" düşük ise, o halde Yazılım Riski düşüktür.

4-Eğer "yazılım projesindeki gereksinimlerin sabitlik seviyesi" yüksek ve "geliştirilecek yazılım ürününün yenilik düzeyi" düşük ve "yazılım ekibinde çalışanların konuştukları dil farklılığı" düşük ve "yazılım ekibinde çalışanların kültür farklılığı" düşük ise, o halde Yazılım Riski düşüktür.

5-Eğer "yazılım ekibinde çalışanların ortak tecrübe seviyesi" yüksek ve "yazılım geliştirme sürecinde yapılacak işlerin birbirine bağlılığı" düşük ise, o halde Yazılım Riski düşüktür.

6-Eğer "yazılım ekibinde çalışanların geliştirilecek ürünle ilgili bilgisi" yüksek ve "yazılım geliştirme sürecinde çalışanlar arasındaki iletişim düzeyi" yüksek ise, o halde Yazılım Riski düşüktür.

7-Eğer "yazılım geliştirme sürecindeki üretkenlik seviyesi" yüksek ve "yazılım geliştirme sürecindeki iletişim ağının etkinliği" yüksek ise, o halde Yazılım Riski düşüktür.

8-Eğer "yazılım geliştirme sürecinde uygun olmayan geliştirme yöntemi kullanımı" düşük ve "yazılım geliştirme sürecinde etkin olmayan iletişim ağı" düşük ve "yazılım projesi ile ilgili açık/net olmayan ya da yanlış anlaşılan gereksinimlerin seviyesi" düşük ise, o halde Yazılım Riski düşüktür.

9-Eğer "proje geliştirme sürecinde yapılacak işlerin belirsizlik seviyesi" düşük ve "etkin proje yönetiminin eksiklik seviyesi" düşük ise, o halde Yazılım Riski düşüktür.

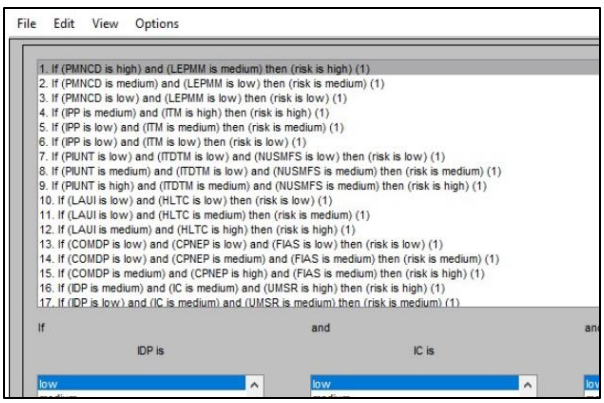
10-Eğer "proje geliştirme sürecinde uygun olmayan proje planlaması" düşük ve "proje ekibinde çalışanların deneyimsizlik seviyesi" düşük ise, o halde Yazılım Riski düşüktür.

11-Eğer "geliştirilecek yazılım projesinde yeni ortaya çıkan teknoloji kullanımı" düşük ve "proje ekibindeki yeterli eğitimi olmayan kişilerin sayısı" düşük ve "geliştirilecek sistemde yeni ya da tanıdık olmayan noktaların veya hususların sayısı" düşük ise, o halde Yazılım Riski düşüktür.

12-Eğer "yazılım geliştirme sürecinde kullanıcı görüşlerinin eksikliği" düşük ve "yazılım geliştirme sürecinde ortaya çıkan teknik karmaşıklık seviyesi" düşük ise, o halde Yazılım Riski düşüktür.

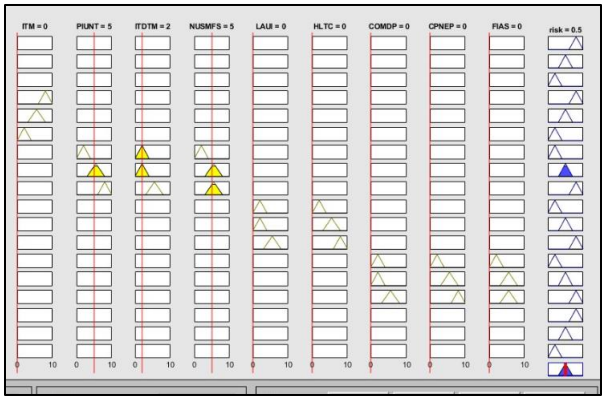
13-Eğer "yazılım geliştirme sürecinde proje ile ilgili değişiklik seviyesi" düşük ve "yazılım geliştirme sürecinde izlenen politikanın negatif etkisi" düşük ve "yazılım geliştirme sürecinde var olan paydaşların tanımının/tanıtımının yetersiz olması" düşük ise, o halde Yazılım Riski düşüktür.

Yukarıda belirtilen 13 kural gibi, yazılım riskleri kurallar çerçevesinde bulanık mantığa dayalı bulanık yaklaşım yöntemi ile ifade edilip, çeşitli uygulamalar sayesinde çıktıya dönüşebilmektedir. Bu uygulamalardan biri de MATLAB'ın içerisinde barındırdığı "Fuzzy Logic Designer" aplikasyonudur. Bu araç sayesinde rahatlıkla kurallar metne dökülüp gerekli çıktılar alınabilmektedir. Aşağıda gösterilen Şekil 5 ve Şekil 6, MATLAB üzerinde yazılım risklerinin tanınması, belirtilmesi ve ortaya çıkarılması için yazılan kuralları ve bu kurallara bağlı olarak bulanık yaklaşım tekniğiyle elde edilen çıktıları ortaya koyup göstermektedir.



Şekil 5. Bulanık yaklaşımlı kural çıkarımına bağlı yazılım risk tanımı (Birant vd., 2019)

Şekil 5'te yazılım risk tanımlarıyla ilgili çeşitli kurallar eklenmiştir ve bu kurallar İngilizce olarak ifade edilmiştir. Daha sonra bu kurallar uygulamaya sokulup MATLAB üzerinde çeşitli çıktılar elde edilmiştir. Eklenen ve ifade edilen bu kurallardan biri: *"If "PIUNT" (Project involved the use of new technology) is medium and "ITDTM" (Inadequately trained development team members) is low and "NUSMFS" (New and/or unfamiliar subject matter for the system) is medium then risk is medium"* olarak tanımlanmıştır. Ayrıca, belirtilen bu kuralın uygulama ile ortaya çıkan MATLAB sonucu aşağıda Şekil 6'da gösterilmiştir.



Şekil 6. Bulanık yaklaşımlı yazılım risk çıktısı (Birant vd., 2019)

Şekil 6'daki MATLAB çıktısı: *"if "PIUNT" is medium (value: 5) and "ITDTM" is low (value: 2) and "NUSMFS" is medium (value: 5) then risk is medium (value: 0,5)"* anlamına gelip değerlerini ortaya koymaktadır. Bu sonuçla birlikte, MATLAB uygulaması aracılığıyla bulanık yaklaşımın yazılım riskleri üzerinde rol alabileceği görülmektedir.

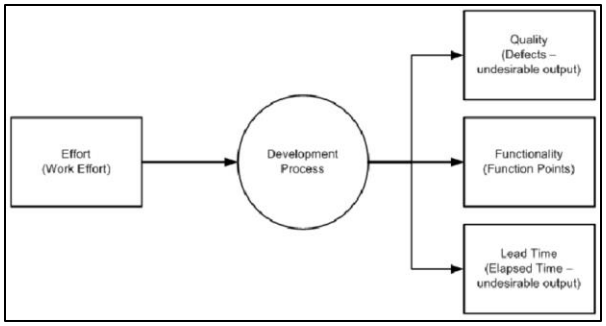
Bütün bu ifade edilenlere ek olarak Şekil 5, MATLAB üzerinde "Fuzzy Logic Designer" aplikasyonunda yazılım risklerinin ortaya çıkarılması için gerekli kuralların yazıldığı, eklendiği, değiştirildiği veya silindiği alan olan "Rule Editor"u göstermektedir. Bununla birlikte Şekil 6 ise, Şekil 5'te metne alınan kuralların yazılım riskleri açısından çıktılarını şekilsel ve grafiksel olarak ifade eden "Rule Viewer"ı göstermektedir.

6. SONUÇ

Yazılım riskleri, belirsiz olayların gerçekleşme olasılığını ve bunların bir organizasyon ya da bir sistem içerisindeki kayıp potansiyelini kapsamaktadır. Kuruluşlar ve firmalar çok teknolojili, çok katmanlı bir ortamda daha fazla uygulama yapmaya devam ettikçe, risk yönetimi yazılım geliştirmenin önemli bir bileşeni haline gelmektedir. Genel mantık olarak yazılım riski, sistem ya da proje geneline yayılan sağlamlık, performans verimliliği, güvenlik, güvenilirlik, fonksiyonellik ve işlem parametrelerini içeren bir risk kombinasyonu olarak görülmektedir. Bununla birlikte yazılım risklerinin ortaya çıkarılması ve tanımlanması oldukça güç bir süreci kapsamaktadır. Bu sürecin faydalı bir şekilde ilerleyebilmesi için etkin bir yöntemin ve yaklaşımın uygulanması gerekmektedir.

"Bulanık Yaklaşım" insan düşünce mekanizmasına uygun çalışma prensibine sahip ve ikili keskin cevaplar (güzel/çirkin, uzun/kısa, zayıf/şişman, kolay/zor, vb.) vermek yerine belirsizliği içinde barındıran gri ve çoklu seçenekler sunma çalışma yapısına sahip yapay zekânın içerisinde yer alan bulanık mantığa dayalı bir tekniktir. Bu çalışma mekanizması sayesinde, belirsizliği ve griliği içinde bulunduran yazılım risklerini tanımlamada ve tanıtmada oldukça etkin bir yöntem olduğu makalede ele alınan yazılım risklerini tanımlayan 13 adet yazılım riskine bağlı kurallar bütünü aracılığıyla açıkça ortaya konmuştur. Buna ek olarak, MATLAB uygulaması sayesinde bulanık yaklaşımın yazılım riskleri konusundaki işlevselliği ve fonksiyonelliği çalışmada gösterilmeye çalışılmıştır. Bu bağlamda, nasıl ki yazılım geliştirme sürecinde temeli oluşturan aşama gereksinimlerin ortaya çıkarılıp tanıtılması ise, yazılım projelerinin başarısını arttırmak için uygulanmaya çalışılan yazılım risk yönetiminde en temel adım yazılım risklerinin ortaya çıkarılıp belirlenmesi ve tanıtılmasıdır. Bu sebepten ötürü, bulanık mantığa dayalı bulanık yaklaşım tekniği, yazılım risk yönetimine katkı vermesi ile birlikte, yazılım projelerinin başarı oranını arttırması açısından oldukça önemli bir yere sahiptir.

Uzman yönetim bilimci Peter DRUCKER tarafından söylendiği iddia edilen bir söze göre, "Ölçemediğiniz süreci değerlendiremezsiniz ve yönetemezsiniz." Bu ifade ışığında, yazılım projelerinin başarısını artırmak için geliştirilen yazılım risk yönetiminde, öncelikli olarak yazılım risklerinin ortaya çıkarılıp tanımlanması gerekmektedir. Bu gereklilik doğrultusunda, çalışmada ele alınan bulanık mantığa dayalı bulanık yaklaşım tekniği, yazılım risklerinin belirlenmesi açısından oldukça güvenilir çıktılar vermektedir. Bu çıktılarla birlikte, yazılım geliştirme sürecinin daha başarılı olabileceği ihtimali ortaya çıkmaktadır. Bu bağlamda, yazılım geliştirmenin ana kaynağı olan insan gücü daha etkin bir şekilde kullanılıp, zaman ve maliyet hesabı açısından şirketler ve kuruluşlar tarafından tasarruflar ve artılar rahatlıkla ortaya çıkabilecektir. Bu iyileştirmenin neticesinde, bulanık yaklaşımlı kural çıkarımına bağlı yazılım risk tanımının etkinliği, "Yazılım Mühendisliği"nin hem bilim alanındaki hem de endüstriyel alandaki rolünün işlevselliğinin elle tutulur, somut kanıtını ortaya açıkça sunabilecektir. Bu bağlamda, yazılım mühendisliği ve yazılım geliştirme sürecinde gerekli olan iş gücü ve kaynaklar arasındaki ilişki aşağıda Şekil 7'de gösterilmeye çalışılmıştır.



Şekil 7. Yazılım mühendisliğinin iş gücü açısından etkisi (Kontio, 2001)

KAYNAKÇA

Arnuphaptrairong, T. (2011). Top ten lists of software project risks: Evidence from the literature survey. *Proceedings of the International MultiConference of Engineers and Computer Scientists, 1,* 1-6.

Birant, K. U., Işık, A. H., Batar, M., Akarsu, H. B., & Tektaş, A. B. (2019). Risk assessment and management method for distributed software development projects with "fuzzy approach". *International Journal of Computer Science and Software Engineering (IJCSSE), 8* (6), 133-139.

Carlsson, C., & Fuller, R. (2003). A fuzzy approach to real option valuation. *Fuzzy Sets and Systems, 139* (2), 297-312.

Dey, P. K., Kinch, J., & Ogunlana, S. O. (2007). Managing risk in software development projects: A case study. *Industrial Management & Data Systems, 107* (2), 284-303.

Fairley, R. (1994). Risk management for software projects. *IEEE Software, 11* (3), 57-67.

Fu, Y., Li, M., & Chen, F. (2012). Impact propagation and risk assessment of requirement changes for software development projects based on design structure matrix. *International Journal of Project Management, 30* (3), 363-373.

Hájek, P. (1998). *Metamathematics of fuzzy logic*. Dordrecht: Kluwer Academic Publishers.

Hall, E. M. (1998). *Managing risk: methods for software systems development*. United Kingdom: Pearson Education

Hoermann, S., Aust, M., Schermann, M., & Krcmar, H. (2012). Comparing risks in individual software development and standard software implementation projects: A delphi study. *2012 45th Hawaii International Conference on System Sciences,* 4884-4893.

Inyang, U. G., & Joshua, E. E. (2013). Fuzzy clustering of students' data repository for at-risks students identification and monitoring. *Computer and Information Science, 6* (4), 37-50.

Jiang, J., & Klein, G. (2000). Software development risks to project effectiveness. *The Journal of Systems and Software, 52* (1), 3-10.

Karataş, F., Koyuncu, İ., Tuna, M., ve Alçın, M. (2020). Bulanık mantık üyelik fonksiyonlarının fpga üzerinde gerçeklenmesi. *Bilgisayar Bilimleri ve Teknolojileri Dergisi 1* (1) , 1-9.

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. (1998). A framework for identifying software project risks. *Communications of the ACM, 41* (11), 76-83.

Kontio, J. (2001). *Software engineering risk management: a method, improvement framework, and empirical evaluation.* PhD Thesis, Helsinki University of Technology, Espoo.

Lezzoni, L. K. (1997). The risks of risk adjustment. *JAMA Journal of the American Medical Association, 278* (19), 1600-1607.

Renn, O. (2004). Perception of risks. *Toxicology Letters, 149* (1), 405-413.

Ross, T. J. (2017). *Fuzzy Logic with engineering applications.* United Kingdom: John Wiley & Sons Inc.

Shen, Q., & Chouchoulas, A. (2002). A rough-fuzzy approach for generating classification rules. *Pattern Recognition, 35* (11), 2425-2438.

Smith, M. (1989). The people risks. *Computer Law & Security Review, 4* (6), 2-6.